User's Manual
V3.2

Micriμm
Press

Weston, FL 33326

Micriμm
1290 Weston Road, Suite 306
Weston, FL 33326
USA

www.micrium.com

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where Micriμm Press is aware of a trademark claim, the product name appears in initial capital letters, in all capital letters, or in accordance with the vendor's capitalization preference. Readers should contact the appropriate companies for more complete information on trademarks and trademark registrations. All trademarks and registered trademarks in this manual are the property of their respective holders.

Copyright © 2013 by Micriμm except where noted otherwise. All rights reserved. Printed in the United States of America. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

μC/Probe and the accompanying files are sold "as is". Micriμm makes and customer receives from Micriμm no express or implied warranties of any kind with respect to the software product, documentation, maintenance services, third party software, or other services. Micriμm specifically disclaims and excludes any and all implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Due to the variety of user expertise, hardware and software environments into which μC/Probe may be subjected, the user assumes all risk of using μC/Probe. The maximum liability of Micriμm will be limited exclusively to the purchase price.

600-uC-Probe-004

Micriμm Press

Table of Contents

Chapter 1 Introduction 6

Chapter 2 µC/Probe System Overview 9
2-1 µC/Probe Data Client 11

Chapter 3 µC/Probe Symbol Browser 15
3-1 Loading an ELF file 15
3-2 Browsing the ELF file 16
3-3 Browsing Chip Definition Files 18

Chapter 4 µC/Probe Settings 19
4-1 General Settings 20
4-2 Communication Settings 21
4-2-1 Segger J-Link 23
4-2-2 USB 24
4-2-3 TCP/IP 25
4-2-4 RS-232 26

Chapter 5 µC/Probe Workspace Explorer 27

Chapter 6 µC/Probe Toolbox 29
6-1 Writable Controls 30
6-2 Linear Gauges 31
6-3 Horizontal Linear Gauges 31
6-4 Quadrant Gauges 32
6-5 Semicircle Gauges 32
6-6 Circular Gauges 33
6-7 Half Donuts 33

6-8 Cylinders 34
6-9 Charts 34
6-10 Numeric Indicators 35
6-11 Advanced 35

Chapter 7 µC/Probe Layout Design Tools 36
7-1 µC/Probe Example 38

Chapter 8 Associating Symbols to Virtual Controls and Indicators 39

Chapter 9 Run-Time Mode 42
9-1 Run-Time Checklist 42
9-2 Running µC/Probe and your Debugging Software at the same time 43
9-3 IAR Systems C-SPY Plugin for µC/Probe 45
9-3-1 Configuring the TCP/IP Bridge between IAR C-SPY and µC/Probe . 46

Appendix A Configuring Virtual Controls and Indicators 47
A-1 Virtual Indicators 48
A-2 Virtual Controls 52
A-3 Charts 60

Appendix B Kernel Awareness Screen 69

Appendix C Terminal Window Control 72
C-1 Terminal Window Control Configuration 74
C-2 Properties Editor 75

Appendix D µC/Trace Triggers Control 76

Appendix E Spreadsheet Control 79
E-1 Adding an instance of the Spreadsheet Control 80
E-2 Configuring the Spreadsheet 81
E-3 Other Features 82
E-4 Application Example 83

Appendix F Scripting Control 84
F-1 Writing a Script 84
F-2 Adding an Instance of the Scripting Control 88
F-3 Configuring the Scripting Control 89
F-4 Executing the Script 90

Appendix G Licensing 92
G-1 Ordering 93
G-2 Activating 94

Appendix H Bibliography 96

Index 97

Chapter

1

Introduction

µC/Probe is a Windows application designed to read and write the memory of any embedded target processor during run-time. Memory locations are mapped to a set of virtual controls and indicators placed on a dashboard. Figure 1-1 shows an overview of the system and data flow.

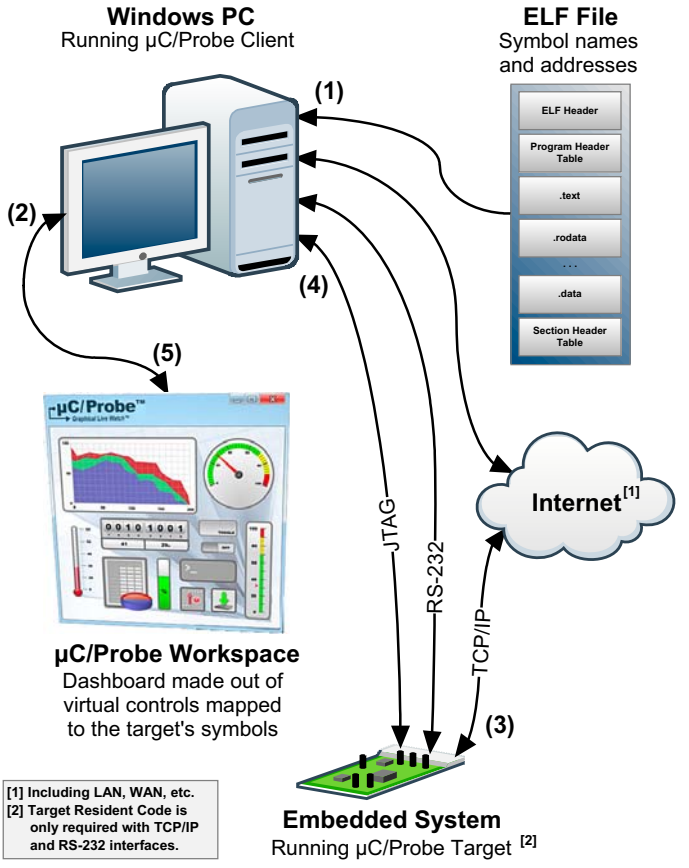


Figure 1-1 **µC/Probe Data Flow Diagram**

F1-1(1) You have to provide µC/Probe with an ELF file with DWARF-2, -3 or -4 debugging information. The ELF file is generated by your toolchain's linker. µC/Probe parses the ELF file and reads the addresses of each of the embedded target's symbols (i.e. global variables) and creates a catalog known as *symbol browser*, which will be used by you during design-time to select the symbols you want to display on your dashboard. Refer to the document µC/Probe Target Manual for more information on installing the µC/Probe Target C files and building the ELF file.

Alternatively, you can also provide a chip definition file that contains the chip's peripheral register addresses or provide your own custom XML based symbol file for those cases when your toolchain cannot generate one of the supported ELF formats.

F1-1(2) During design-time, you create a µC/Probe workspace using a Windows PC and µC/Probe. You design your own dashboard by dragging and dropping virtual controls and indicators onto a *data screen*. Each virtual control and indicator needs to be mapped to an embedded target's symbol by selecting it from the symbol browser. This document aims at providing more information on creating your own dashboard with µC/Probe.

F1-1(3) Before proceeding to the run-time stage, µC/Probe needs to be configured to use one of the four communication interfaces: JTAG, USB, RS232 or TCP/IP. In order to start the run-time stage, you click the *Play* button and µC/Probe starts making requests to read the value of all the memory locations associated with each virtual control and indicator (i.e. buttons and gauges respectively). At the same time, µC/Probe sends commands to write the memory locations associated with each virtual control (i.e. buttons on a click event).

F1-1(4) In the case of a reading request, the embedded target responds with the latest value. In the case of a write command, the embedded target responds with an acknowledgement. Refer to the document µC/Probe Target Manual for more information on all you need in regards to the firmware that implements the communication interface that runs on the embedded target.

F1-1(5) µC/Probe parses the responses from the embedded target and updates the virtual controls and indicators.

Refer to the document µC/Probe Target Manual for more information about the firmware that resides on the Embedded System.

This document only provides information about the Windows PC side of the system.

Chapter

2

µC/Probe System Overview

This section provides an overview of the µC/Probe Windows Application.

Whenever you start µC/Probe in your Windows PC, three different modules are started: µC/Probe Automatic Updates and Licensing System, µC/Probe Data Client and µC/Probe Generic Target Communications Module as illustrated in Figure 2-1:

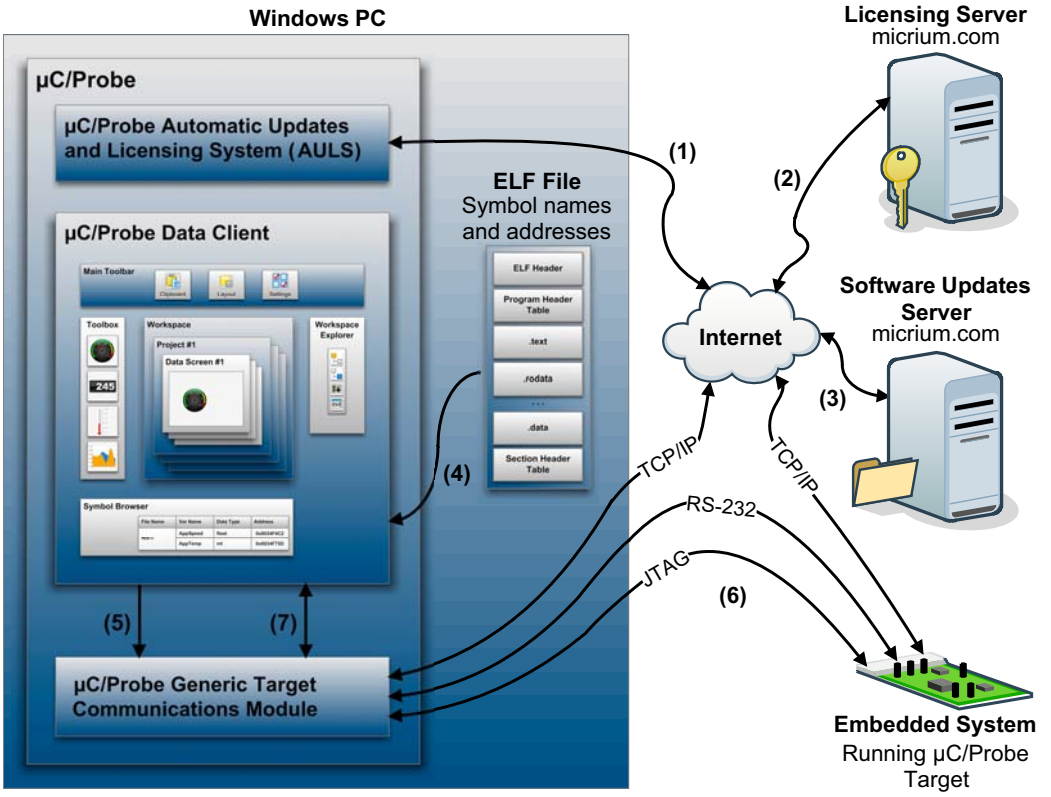


Figure 2-1 **µC/Probe System Overview**

F2-1(1) The Automatic Updates and Licensing System (AULS) is the part of the application that allows you to install and keep your µC/Probe application up to date.

F2-1(2) The Educational Edition of µC/Probe is deployed with some of the Basic and Professional Editions' features and do not require internet access to activate the software application. The Basic and Professional Edition of µC/Probe require internet access to validate the license key provided by your Micriμm's sales representative.

For more information on µC/Probe Licensing see Appendix G, "Licensing" on page 92.

F2-1(3) All Editions of µC/Probe are self-updating and every time you start the application, if internet access is available, the µC/Probe AULS module checks for newer versions of µC/Probe from the Micriμm website and as they become available, the µC/Probe AULS module, automatically replaces any updated files.

F2-1(4) The µC/Probe Data Client is the part of the application that allows you to design your dashboard (design-time mode) and run it (run-time mode).

The next section in this document provides more information in regards to using the µC/Probe Data Client during design-time and run-time.

F2-1(5) The µC/Probe Generic Target Communications Module is the part of the application that connects directly with the Embedded Target and responds to the requests from the Data Client.

When the µC/Probe run-time mode gets started, the Data Client sends requests to the Generic Target Communications Module. The requests contain not only the embedded target communication settings but also all the symbol's memory address required by your dashboard design.

F2-1(6) The Generic Target Communications Module takes the request from the µC/Probe Data Client and initiates a communication with the embedded target through the configured communication interface.

F2-1(7) The µC/Probe Data Client exchanges requests to read and write the memory locations required by the current view of your dashboard's design with the embedded target through the Generic Target Communications Module.

2-1 µC/PROBE DATA CLIENT

The µC/Probe Data Client is illustrated in more detail in Figure 2-2:

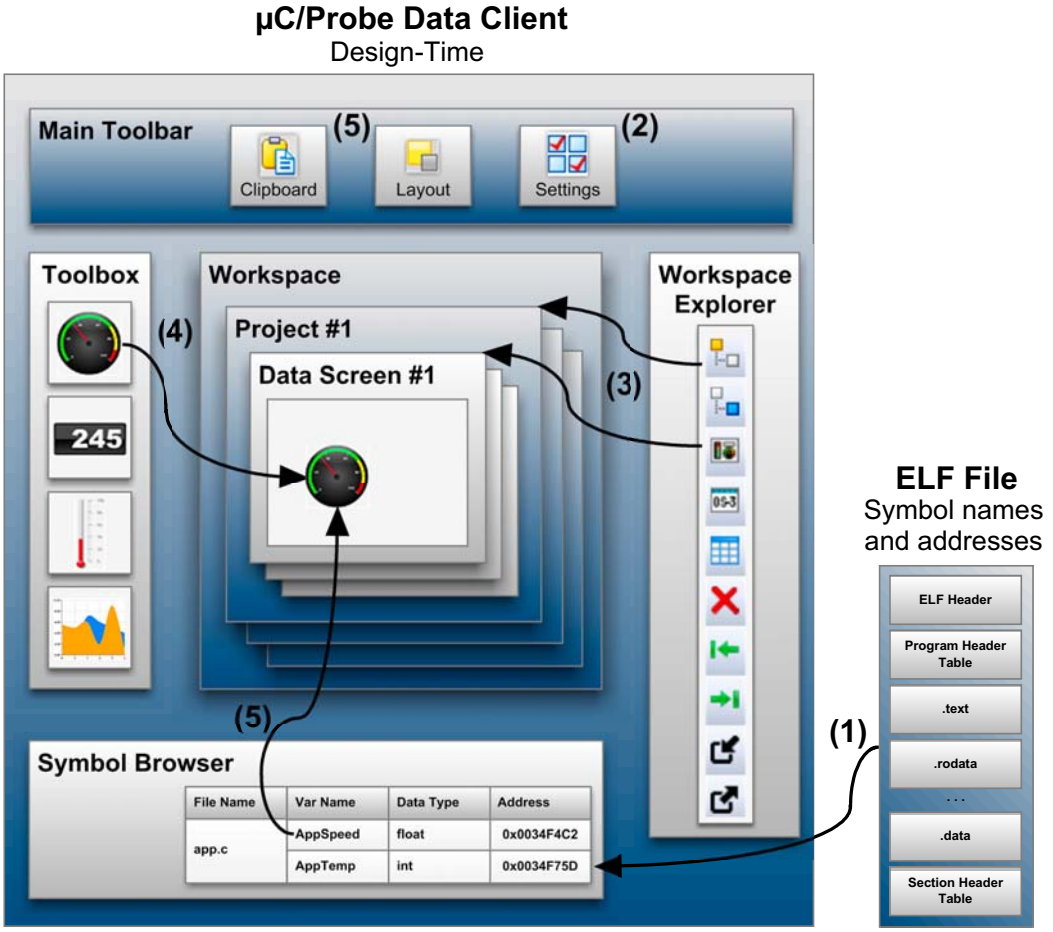


Figure 2-2 **µC/Probe Data Client: Design Time**

F2-2(1) The µC/Probe Data Client is the part of the application that during design-time takes the ELF file with DWARF-2, -3 or -4 debugging information. As previously discussed, the ELF file is generated by your toolchain's linker. The µC/Probe Data Client parses the ELF file and reads the addresses of each of the embedded target's symbols (i.e. global variables) and creates a catalog known as symbol browser, which will be used by you during design-time to select the symbols you want to display on your dashboard. Refer to the document µC/Probe Target Manual for more information on installing the µC/Probe Target C files and building the ELF file.

For more information, see Chapter 3, "µC/Probe Symbol Browser" on page 15.

F2-2(2) During design-time it is necessary to adjust the communication and other general settings. See Chapter 4, "µC/Probe Settings" on page 19 for more information on configuring µC/Probe.

F2-2(3) The Workspace Explorer in the µC/Probe Data Client allows you to add or delete Projects and Data Screens among other things.

For more information, see Chapter 5, "µC/Probe Workspace Explorer" on page 27.

F2-2(4) The µC/Probe Toolbox displays icons for the virtual controls and indicators that you can add to your Data Screens. Each toolbox icon can be dragged and dropped onto the Data Screen to build your own dashboard.

For more information, see Chapter 6, "µC/Probe Toolbox" on page 29.

F2-2(5) The µC/Probe Layout Design Tools help you arrange the virtual controls and indicators on your data screen by speeding up the creation of your dashboard and making it look great.

For more information, see Chapter 7, "µC/Probe Layout Design Tools" on page 36.

F2-2(6) The last step during design-time is to map each virtual control and indicator in your Data Screen with an Embedded Target's memory location. The symbol browser allows you to quickly find the variable you want to display and then all you have to do is drag the variable from the symbol browser and drop it onto the virtual control or indicator of your choice.

See Chapter 8, "Associating Symbols to Virtual Controls and Indicators" on page 39 for more information on using the symbol browser to map virtual controls and indicators to the embedded target's memory locations.

The actual µC/Probe windows application is shown in Figure 2-3:

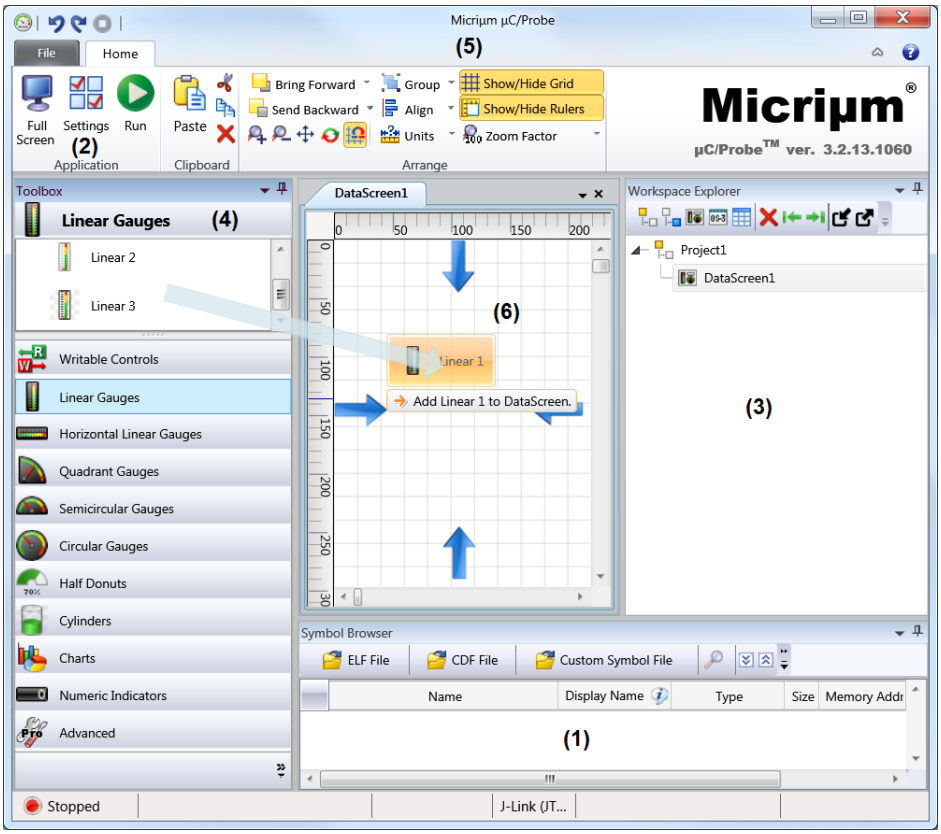


Figure 2-3 **µC/Probe Windows Application**

F2-3(1) Chapter 3, "µC/Probe Symbol Browser" on page 15.

F2-3(2) Chapter 4, “µC/Probe Settings” on page 19.

F2-3(3) Chapter 5, “µC/Probe Workspace Explorer” on page 27.

F2-3(4) Chapter 6, “µC/Probe Toolbox” on page 29.

F2-3(5) Chapter 7, “µC/Probe Layout Design Tools” on page 36.

F2-3(6) Chapter 8, “Associating Symbols to Virtual Controls and Indicators” on page 39.

Chapter

3

µC/Probe Symbol Browser

The µC/Probe's Symbol Browser is a list of your embedded target's symbols that helps you quickly find the symbol you want to use in your data screen. The symbol browser is available during design-mode and it is located at the bottom of the application window.

3-1 LOADING AN ELF FILE

In order to provide µC/Probe with the path of the embedded target's output file (ELF file), locate and click the buttons indicated in Figure 3-1:

Figure 3-1 **µC/Probe Symbol Browser: Loading an ELF file**

µC/Probe will prompt for the ELF file path by using a standard open file dialog. Locate the output file in your PC. The path is usually configured from your toolchain's linker configuration.

The ELF file needs to have symbolic information for debugging purposes in the DWARF-2, -3 or -4 format.

Alternatively, you can also provide a chip definition file that contains the chip's peripheral register addresses or provide your own custom XML based symbol file for those cases when your toolchain cannot generate one of the supported ELF formats.

Refer to the document µC/Probe-Target Manual for more information on building and configuring your toolchain to generate an ELF file or create your own XML based custom symbol file.

3-2 BROWSING THE ELF FILE

µC/Probe parses the ELF file and creates a catalog that you can browse to search for the symbol you need. The symbol browser is a five-column tree list of symbols grouped by the C file name where the variable was declared as shown in Figure 3-2:

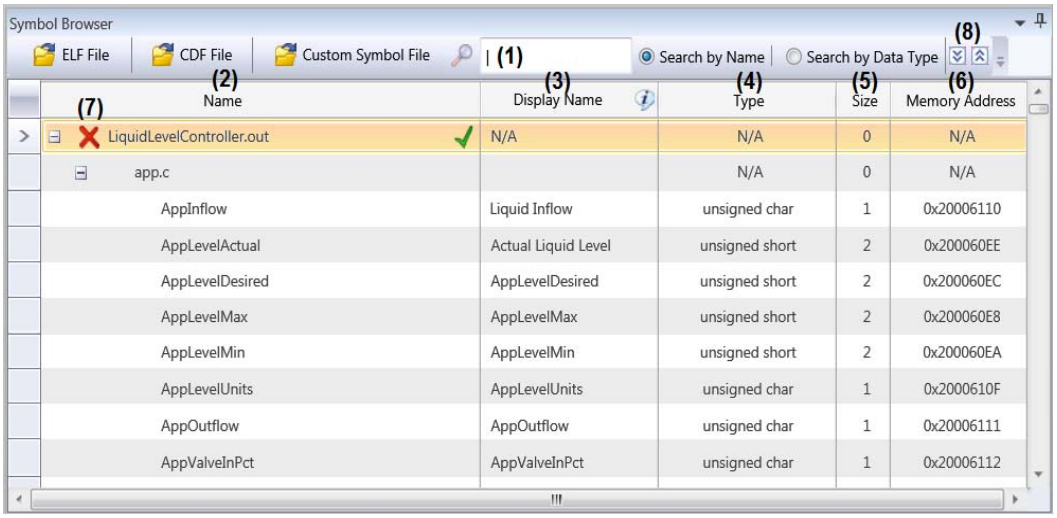


Figure 3-2 **µC/Probe Symbol Browser: Symbols grouped by C file**

F3-2(1) The symbol browser allows you to quickly find the symbol you want. Click on the symbol browser headers row to sort the list by the column you want. You can also expand and collapse tree nodes to focus on a particular C file, or you can use the search box and search by symbol name or data type.

F3-2(2) The **Name** column shows the name of the symbol as declared in your C file.

F3-2(3) The **Display Name** column by default displays the name of the symbol as declared in your C file, but also allows you to create an alias for the symbol. Double-click over the Display Name cell to create an alias.

F3-2(4) The **Type** column displays the symbol's C data type.

F3-2(5) The **Size** column displays the size in bytes of the symbol.

F3-2(6) The **Memory Address** column displays the symbol's location in the embedded target's memory.

F3-2(7) Click on the red **X** next to the name of the ELF file, to remove a symbol file from the symbol browser.

F3-2(8) The *expand all* and *collapse all* buttons allow you to browse more efficiently throughout the symbol browser tree.

Be aware that the symbol browser in µC/Probe will detect if the ELF file has been re-compiled and will refresh all addresses. However, in case you move the ELF file to a different location in your file system, µC/Probe cannot update the addresses automatically. Instead, you can update the symbol browser with the new path by first removing the ELF file (red X next to the filename) and then opening the new ELF file.

3-3 BROWSING CHIP DEFINITION FILES

µC/Probe gets installed with a large catalog of *Chip Definition Files* that includes thousands of chips. You can browse the catalog and select your platform's chip definition file by clicking the button *Open CDF File* and using the CDF browser shown in Figure 3-3:

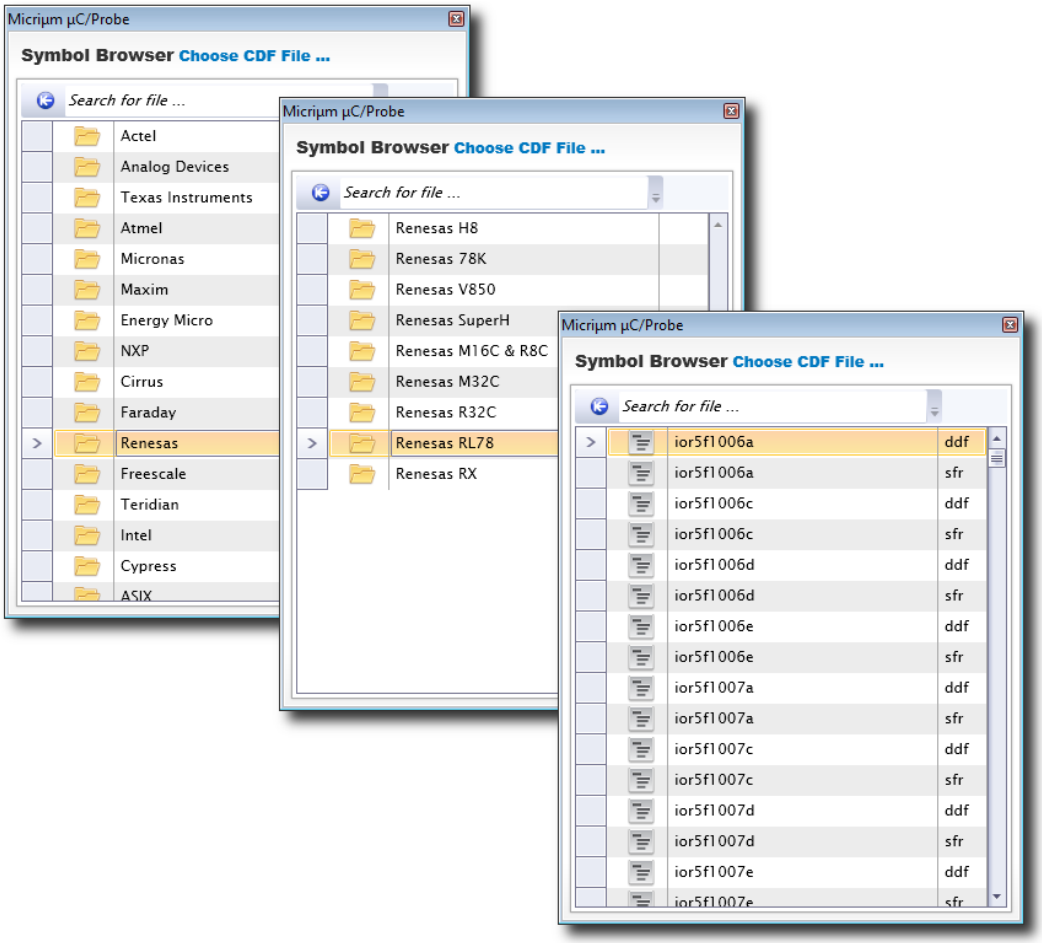


Figure 3-3 **Chip Definition Files Browser**

The CDF files provide the chip's peripheral I/O register names and addresses which you can use to associate with controls such as the *Bit Control* described in section A-2-8 "Bit Control Properties Editor" on page 59.

Chapter

4

µC/Probe Settings

The µC/Probe application's tool bar is located at the top of the application window.

The µC/Probe Settings window is opened by making click on the **Settings** button in the application's tool bar as indicated in Figure 4-1:

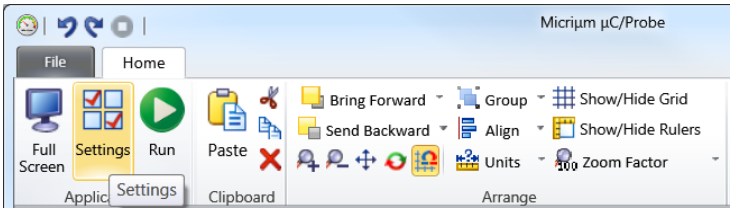


Figure 4-1 **µC/Probe Toolbar: Settings**

The settings window is divided in the following categories:

- General Settings:

  The general settings include the application debug logging, automatic updates and data collection settings.

- Communication Settings:

  The communication settings window includes the endianness type and the communication interface settings.

4-1 GENERAL SETTINGS

Figure 4-2 shows the µC/Probe General Settings window:

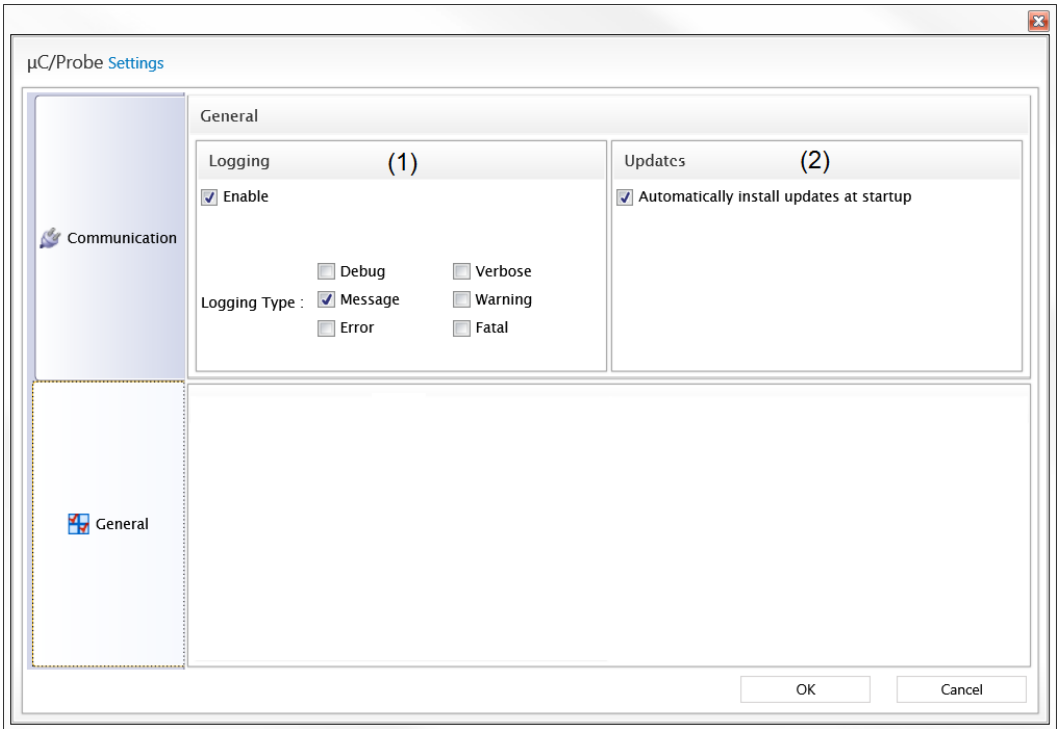


Figure 4-2 **µC/Probe General Settings**

F4-2(1) µC/Probe can be configured to log a specific level of verbosity for technical support purposes. If you ever have to contact Micriµm's technical support for any issues with your µC/Probe application, you can select a logging type that better describes your failure scenario.

F4-2(2) µC/Probe can be configured to automatically check and install updates as they become available from the Micriµm website. µC/Probe will check for software updates at startup if internet access is available.

4-2 COMMUNICATION SETTINGS

µC/Probe supports a variety of communication interfaces. See the communication options advisor in Figure 4-3 to select the best option for your platform and application:

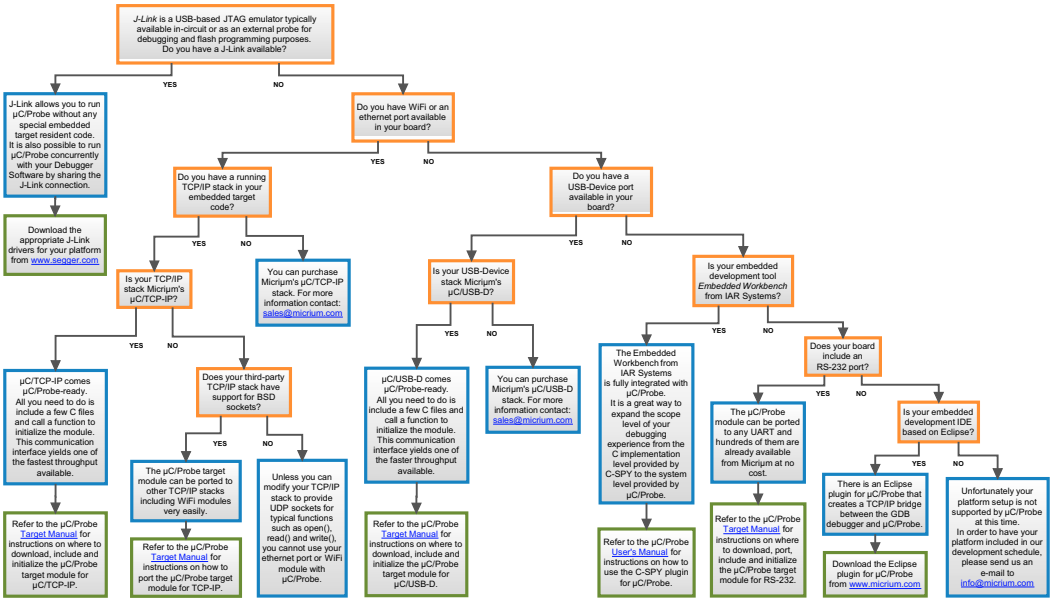

J-Link is a USB-based JTAG emulator typically available in-circuit or as an external probe for debugging and flash programming purposes. Do you have a J-Link available?
YES
NO
J-Link allows you to run µC/Probe without any special embedded target resident code. It is also possible to run µC/Probe concurrently with your Debugger Software by sharing the J-Link connection.
Download the appropriate J-Link drivers for your platform from www.segger.com
Do you have WiFi or an ethernet port available in your board?
YES
NO
Do you have a running TCP/IP stack in your embedded target code?
YES
NO
Is your TCP/IP stack Micrium's µC/TCP-IP?
YES
NO
You can purchase Micrium's µC/TCP-IP stack. For more information contact: sales@micrium.com
µC/TCP-IP comes µC/Probe-ready. All you need to do is include a few C files and call a function to initialize the module. This communication interface yields one of the fastest throughput available.
Refer to the µC/Probe Target Manual for instructions on where to download, include and initialize the µC/Probe target module for µC/TCP-IP.
Does your third-party TCP/IP stack have support for BSD sockets?
YES
NO
The µC/Probe target module can be ported to other TCP/IP stacks including WiFi modules very easily.
Refer to the µC/Probe Target Manual for instructions on how to port the µC/Probe target module for TCP-IP.
Unless you can modify your TCP/IP stack to provide UDP sockets for typical functions such as open(), read() and write(), you cannot use your ethernet port or WiFi module with µC/Probe.
Do you have a USB-Device port available in your board?
YES
NO
Is your USB-Device stack Micrium's µC/USB-D?
YES
NO
µC/USB-D comes µC/Probe-ready. All you need to do is include a few C files and call a function to initialize the module. This communication interface yields one of the faster throughput available.
Refer to the µC/Probe Target Manual for instructions on where to download, include and initialize the µC/Probe target module for µC/USB-D.
You can purchase Micrium's µC/USB-D stack. For more information contact: sales@micrium.com
Is your embedded development tool Embedded Workbench from IAR Systems?
YES
NO
The Embedded Workbench from IAR Systems is fully integrated with µC/Probe. It is a great way to expand the scope level of your debugging experience from the C implementation level provided by C-SPY to the system level provided by µC/Probe.
Refer to the µC/Probe User's Manual for instructions on how to use the C-SPY plugin for µC/Probe.
Does your board include an RS-232 port?
YES
NO
The µC/Probe module can be ported to any UART and hundreds of them are already available from Micrium at no cost.
Refer to the µC/Probe Target Manual for instructions on where to download, port, include and initialize the µC/Probe target module for RS-232.
Is your embedded development IDE based on Eclipse?
YES
NO
There is an Eclipse plugin for µC/Probe that creates a TCP/IP bridge between the GDB debugger and µC/Probe.
Download the Eclipse plugin for µC/Probe from www.micrium.com
Unfortunately your platform setup is not supported by µC/Probe at this time. In order to have your platform included in our development schedule, please send us an e-mail to info@micrium.com

Figure 4-3 **Communication Options Advisor**

Once you have chosen the appropriate communication interface for your platform, you can configure µC/Probe from the communication settings window shown in Figure 4-4:

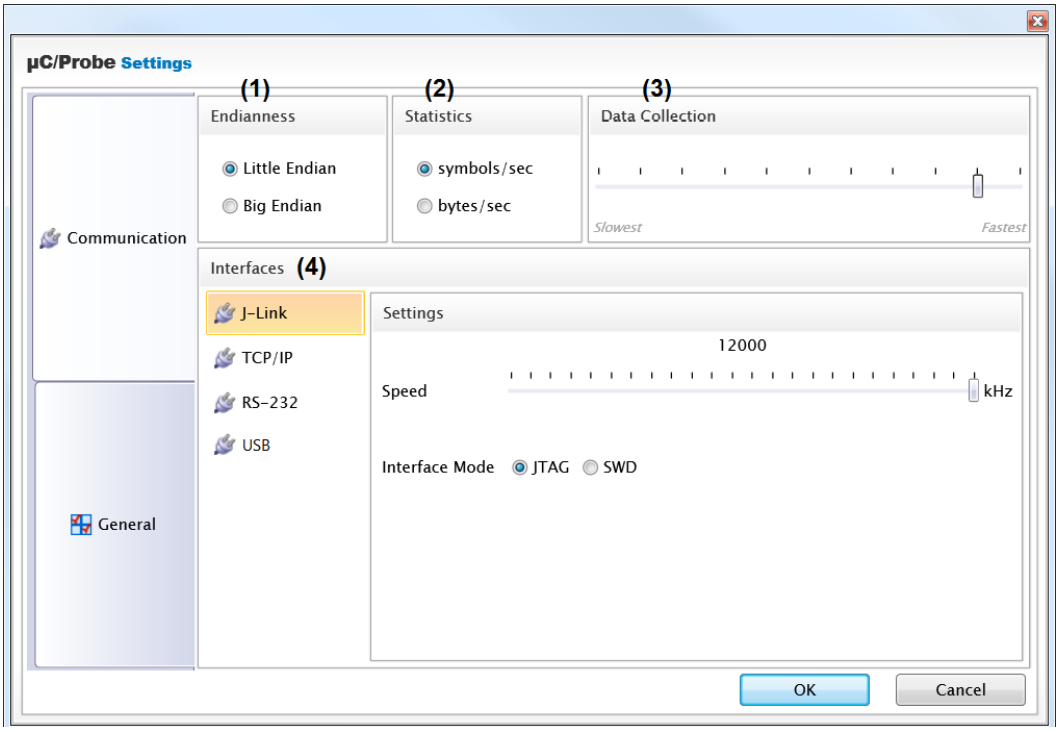


Figure 4-4 **µC/Probe Communication Settings**

F4-4(1) µC/Probe can be configured to interpret the byte order in either little or big endian, select the option that corresponds to your target's endianness type.

F4-4(2) During run-time, µC/Probe calculates the speed of your connection and displays the value in either symbols per second or bytes per second.

F4-4(3) You can adjust how fast you want µC/Probe to make requests to the target depending on your application's bandwidth. The fastest collection rate takes a toll in both the embedded target and your Windows PC resources.

F4-4(4) µC/Probe supports J-Link, USB, TCP/IP and RS-232. Select the interface that your target supports, and configure the settings corresponding to the interface.

The following sections describe how to configure each communication interface. Refer to the document µC/Probe-Target Manual for more information on the communication interface supported by the embedded target.

4-2-1 SEGGER J-LINK

J-Link is a USB powered JTAG emulator designed by Segger. In order to install the windows drivers for J-Link (J-Link DLL) go to Segger's website at www.segger.com and download the J-Link software pack for Windows.

J-Link is the most popular emulator for ARM cores and it does not require any special code resident in the embedded target to connect with µC/Probe.

If using J-Link, you can interface µC/Probe even with a bare-metal application running no kernel at all, as shown in Figure 4-5:

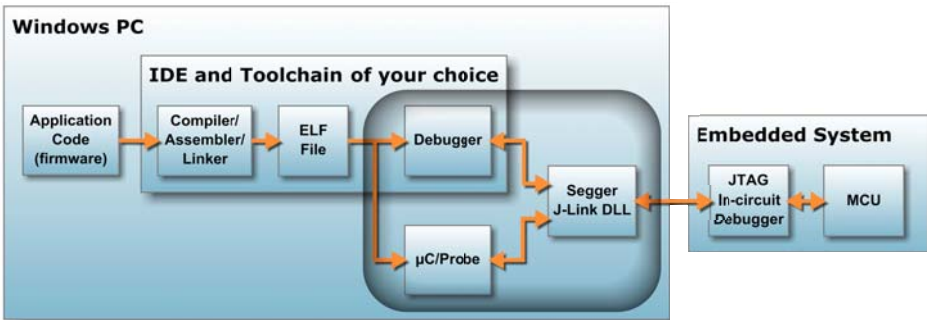


Figure 4-5 **µC/Probe via J-Link**

µC/Probe supports two types of J-link interface modes. Select the interface mode from the radio buttons shown in Figure 4-6:

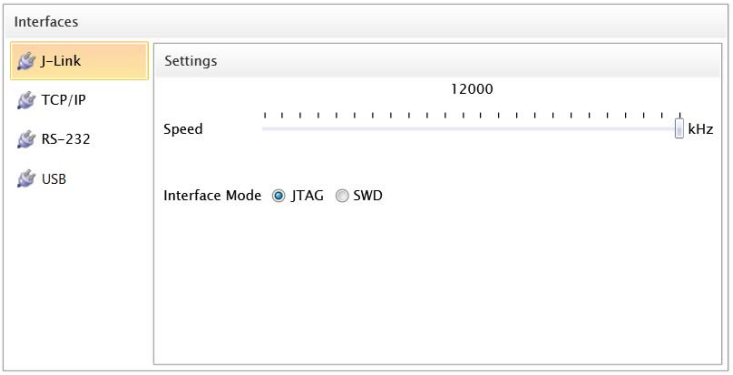


Figure 4-6 **µC/Probe Communication Settings: J-Link**

4-2-2 USB

µC/Probe supports a USB interface over the µC/USB Device stack by Micriµm. This USB interface as opposed to J-Link, requires µC/Probe-Target code resident in your embedded system and because of the nature of USB, it also requires a kernel. Micriµm supports many cores and most likely this code is available from Micriµm. Contact Micriµm to find out if resident code for your particular setup is available.

Once your embedded system is running µC/Probe-Target as described in the document µC/Probe-Target Manual, the device should be ready to connect after plugging in. The Windows computer will enumerate the device and will display it as one of the available devices in the communication settings window as shown in Figure 4-7:

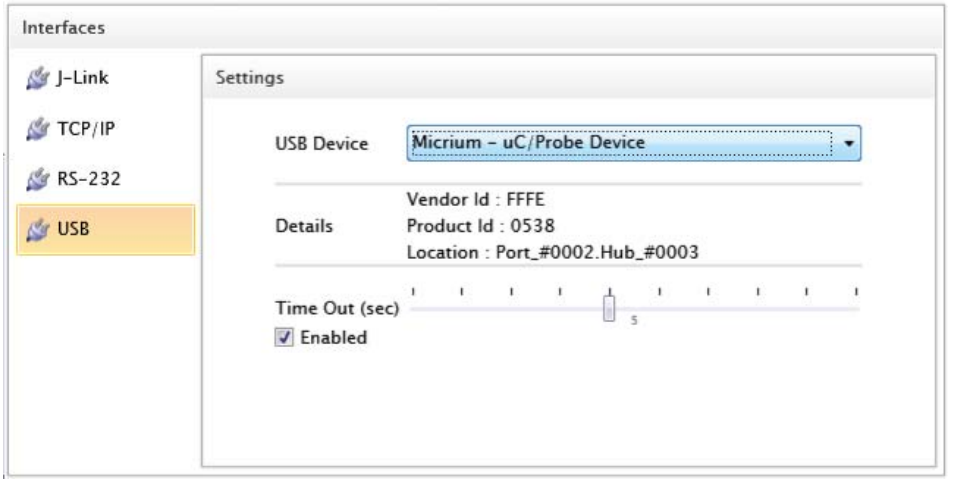


Figure 4-7 **µC/Probe Communication Settings: USB**

If you have more than one µC/Probe USB Device you can distinguish it by the information on the *Details* pane right below the drop-down box.

You can specify an optional timeout in seconds, which is the time µC/Probe is willing to wait for the target to respond before presenting an error message.

4-2-3 TCP/IP

µC/Probe supports a TCP/IP interface over the UDP protocol. This TCP/IP interface as opposed to J-Link, requires µC/Probe-Target code resident in your embedded system and because of the nature of TCP/IP, it also requires a kernel. Micriμm supports many cores and most likely this code is available from Micriμm. Contact Micriμm to find out if resident code for your particular setup is available.

Assuming your embedded system is running µC/Probe-Target as described in the document µC/Probe-Target Manual, enter the IP address and port number of your embedded system in the text boxes shown in Figure 4-8:

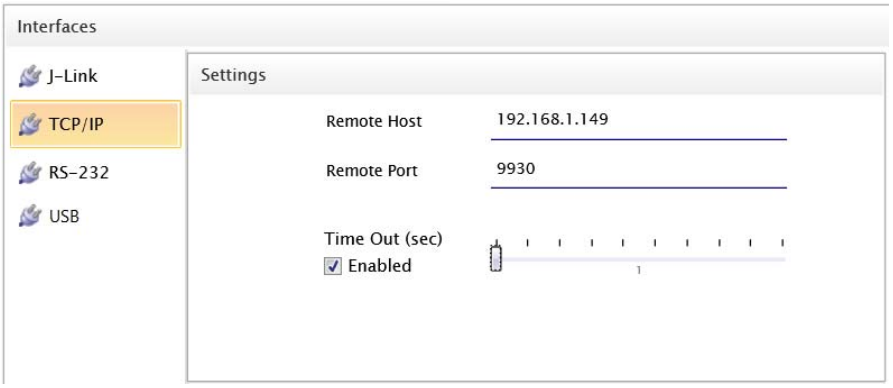


Figure 4-8 **µC/Probe Communication Settings: TCP/IP**

You can specify an optional timeout in seconds, which is the time µC/Probe is willing to wait for the target to respond before presenting an error message.

The TCP/IP interface is also used to interface through third-party plugin proxies such as the IAR Systems C-SPY plugin for µC/Probe as described in section 9-3 “IAR Systems C-SPY Plugin for µC/Probe” on page 45.

4-2-4 RS-232

µC/Probe supports a Serial RS-232 interface. This serial interface as opposed to J-Link, requires µC/Probe-Target code resident in your embedded system. Micriµm supports many UARTs and most likely this code is available from Micriµm. Contact Micriµm to find out if resident code for your particular setup is available.

Assuming your embedded system is running µC/Probe-Target as described in the document µC/Probe-Target Manual, enter the serial COM port number that your embedded target is attached to and select the baud rate from the drop downs shown in Figure 4-9:

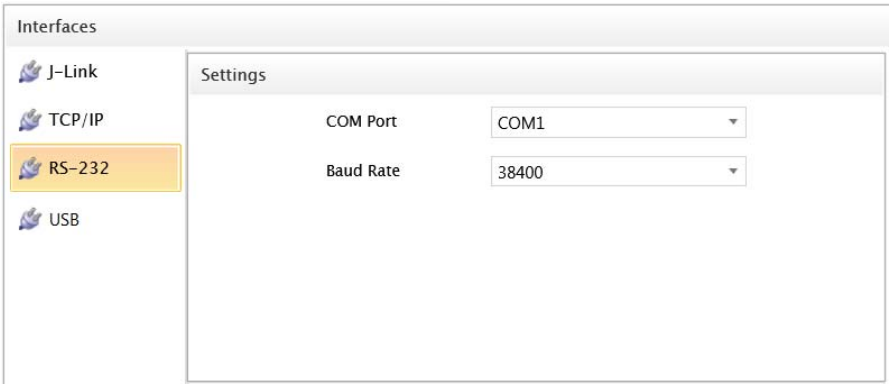


Figure 4-9 **µC/Probe Communication Settings: RS-232**

Chapter

5

µC/Probe Workspace Explorer

The µC/Probe Workspace Explorer is located on the right side of the application window and it is shown in Figure 5-1:

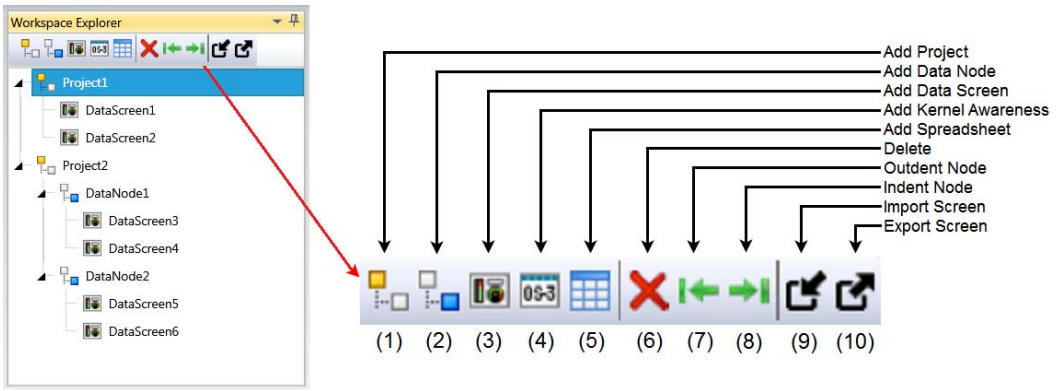


Figure 5-1 **µC/Probe Workspace Explorer**

F5-1(1) µC/Probe allows you to create a dashboard or user interface in a matter of minutes. The data screen is where you drag and drop the virtual controls and indicators. Your data screen's appearance and layout are very important. You can use Projects as a means to separate complex dashboards into multiple regions. Each project can contain multiple data screens and the workspace explorer allows you to navigate through this hierarchy.

F5-1(2) Similar to Projects, Data Nodes are just another level of hierarchy that allows you to group sets of virtual controls and indicators together into categories you define.

F5-1(3) Data Screens are the screens where you drag and drop the virtual controls and indicators. You can add as many data screens as you want.

F5-1(4) The Kernel Awareness Screen is a pre-configured Data Screen with all the symbols related to µC/OS-III. See Appendix B, “Kernel Awareness Screen” on page 69 for more information about this.

F5-1(5) Use this button to create a bridge between µC/Probe and Microsoft Excel.

F5-1(6) Use the Delete button to delete an item from the workspace explorer, including Projects, Data Nodes, Data Screens and Kernel Awareness Screens.

F5-1(7) Use the indent button to push in an item in the Workspace Explorer tree. The items you can adjust the level of indentation include Projects, Data Nodes, Data Screens and Kernel Awareness Screens.

F5-1(8) Use the outdent button to push out an item in the Workspace Explorer tree. The items you can outdent include Projects, Data Nodes, Data Screens and Kernel Awareness Screens.

F5-1(9) Use the Import Screen button to import a previously exported screen.

F5-1(10) Use the Export Screen button to export the screen currently in focus to a file.

In order to organize your workspace tree you can also use your mouse to drag and drop items and rename items by invoking the context menu with a right-click.

Figure 5-2 shows an example of using projects and data nodes to better present a control panel for a liquid level control system:

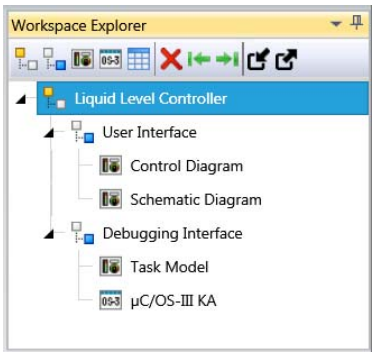


Figure 5-2 **Workspace Explorer for a Liquid Level Control System**

Chapter

6

µC/Probe Toolbox

The µC/Probe Toolbox is located on the left side of the application window and it is shown in Figure 6-1:

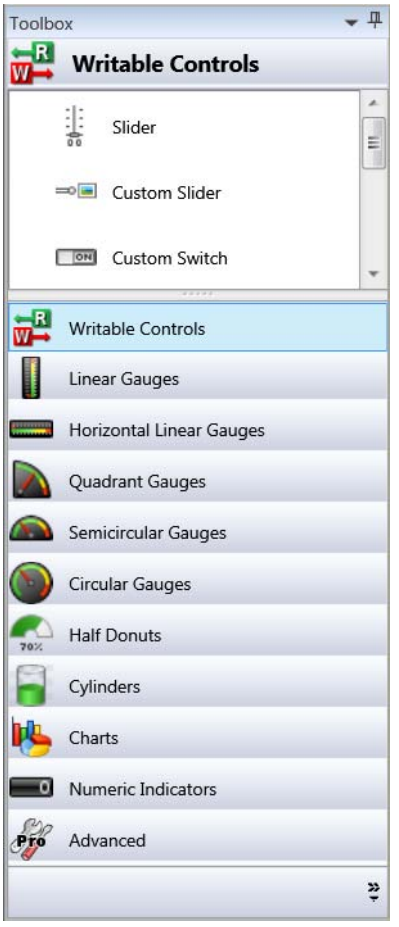


Figure 6-1 **µC/Probe Toolbox**

Depending on the µC/Probe Edition you purchased, the available tools will vary. This document describes all the features found in the Professional Edition of µC/Probe. For more information on which features you have, see Appendix G, "µC/Probe Editions Comparison Table" on page 93.

The items in the toolbox are contained in an accordion type of panel. You click on each button to display the items that belong to a category.

If you are running the Basic or Professional Edition of µC/Probe and have the automatic updates enabled, each category in this toolbox will expand with more virtual controls and indicators as software updates become available.

The following sections present a brief introduction to each of the toolbox categories. For more information on configuring each type of virtual control or indicator, see Appendix A, "Configuring Virtual Controls and Indicators" on page 47.

6-1 WRITABLE CONTROLS

The writable controls shown in Figure 6-2 include buttons, check boxes, sliders and a bit control. Use these controls to modify the value of symbols from the embedded target. For more information configuring the properties of writable controls see Appendix A, "Virtual Controls" on page 52.

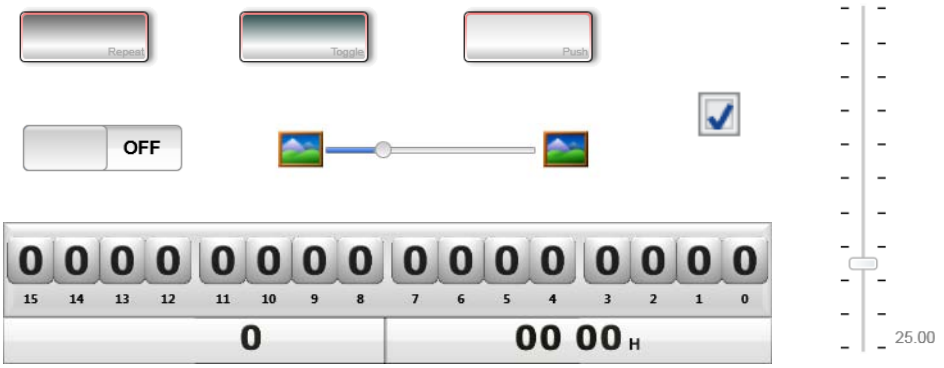


Figure 6-2 **µC/Probe Toolbox: Writable Controls**

6-2 LINEAR GAUGES

Use the linear gauges shown in Figure 6-3 to display numeric data in a tri-color vertical scale. For more information configuring the properties of linear gauges see Appendix A, “Virtual Indicators” on page 48.

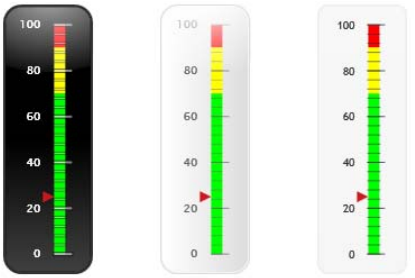


Figure 6-3 **µC/Probe Toolbox: Linear Gauges**

6-3 HORIZONTAL LINEAR GAUGES

Use the horizontal linear gauges shown in Figure 6-4 to display numeric data in a tri-color horizontal scale. For more information configuring the properties of horizontal linear gauges see Appendix A, “Virtual Indicators” on page 48.

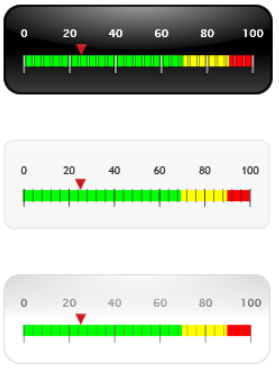


Figure 6-4 **µC/Probe Toolbox: Horizontal Linear Gauges**

6-4 QUADRANT GAUGES

Use the quadrant gauges shown in Figure 6-5 to display numeric data in a tri-color quadrant scale. For more information configuring the properties of quadrant gauges see Appendix A, “Virtual Indicators” on page 48.

Figure 6-5 **µC/Probe Toolbox: Quadrant Gauges**

6-5 SEMICIRCLE GAUGES

Use the semicircle gauges shown in Figure 6-6 to display numeric data in a tri-color semicircular scale. For more information configuring the properties of semicircle gauges see Appendix A, “Virtual Indicators” on page 48.

Figure 6-6 **µC/Probe Toolbox: Semicircle Gauges**

6-6 CIRCULAR GAUGES

Use the circular gauges shown in Figure 6-7 to display numeric data in a tri-color circular scale. For more information configuring the properties of circular gauges see Appendix A, “Virtual Indicators” on page 48.

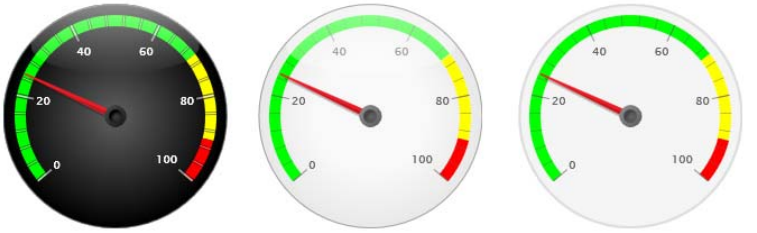


Figure 6-7 **µC/Probe Toolbox: Circular Gauges**

6-7 HALF DONUTS

Use the half donut indicators shown in Figure 6-8 to display numeric data in a bi-color semicircular scale. For more information configuring the properties of half donuts see Appendix A, “Virtual Indicators” on page 48.

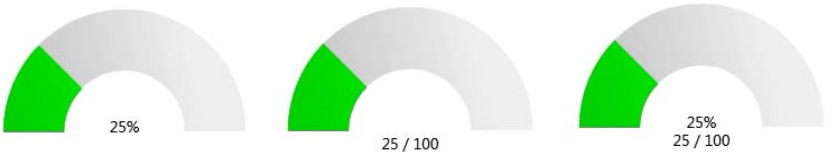


Figure 6-8 **µC/Probe Toolbox: Half Donuts**

6-8 CYLINDERS

Use the cylinder or tank level indicator shown in Figure 6-9 to display numeric data in a solid or gradient color. For more information configuring the properties of cylinders see Appendix A, “Virtual Indicators” on page 48.

Figure 6-9 **µC/Probe Toolbox: Cylinders**

6-9 CHARTS

Use the charts shown in Figure 6-10 to display numeric data including arrays in a marker, line, area or scatter x-y chart. For more information configuring the properties of charts see Appendix A, “Timeline Charts” on page 60.

Figure 6-10 **µC/Probe Toolbox: Charts**

6-10 NUMERIC INDICATORS

Use the numeric indicators shown in Figure 6-11 to display numeric data in text. For more information configuring the properties of numeric indicators see Appendix A, “Formatting Properties Editor” on page 48 and Appendix A, “Numeric Indicator Properties Editor” on page 50.

Figure 6-11 **µC/Probe Toolbox: Numerics**

6-11 ADVANCED

The advanced category of the toolbox includes other miscellaneous indicators such as a text box, terminal window, scripting control, spreadsheet control, µC/Trace trigger control and an image container capable of displaying an indexed array of images. For more information configuring the properties of these advanced controls see Appendix A, “Virtual Indicators” on page 48, Appendix E, “Spreadsheet Control” on page 79 and Appendix F, “Scripting Control” on page 84.

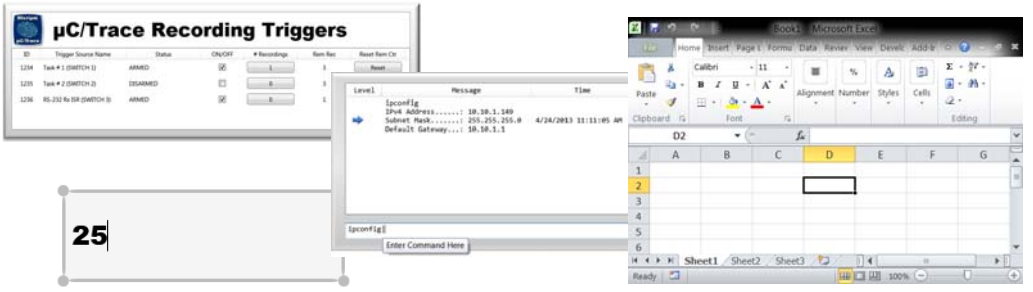


Figure 6-12 **µC/Probe Toolbox: Advanced**

Chapter

7

µC/Probe Layout Design Tools

The Layout Design Tools are located on the Main Toolbar at the top of the application's window. They include tools to arrange the virtual controls and indicators on your data screen as shown in Figure 7-1:

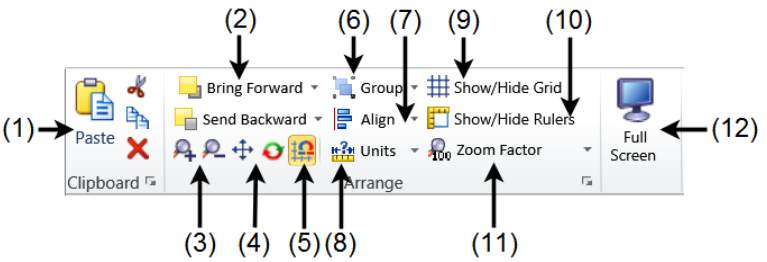


Figure 7-1 **µC/Probe Layout Design Tools**

F7-1(1) µC/Probe allows you to copy, cut and paste a single or multiple virtual controls or indicators. Everything you select will be stored temporarily into µC/Probe's clipboard during your session.

F7-1(2) µC/Probe allows you to drag and drop virtual controls and indicators onto the data screen and have them overlap one another. In some cases may be necessary to move some of them to the front of the data screen. Select the virtual control and indicator you want to move and click the **Bring Forward** or **Send Backward** button to move it to the front or to the back respectively.

F7-1(3) µC/Probe includes accessibility features that make the software more user friendly. You can control the zoom level of your µC/Probe Data Screen during both design-time and run-time. Click the buttons with the magnifier glass to zoom in and out, or use the keyboard shortcuts **Ctrl+** to zoom-in and **Ctrl-** to zoom-out.

F7-1(4) The panning tool in µC/Probe makes it easy to move a data screen around while your are zoomed in. The **Pan** button is a toggle button, clicking the button toggles the panning mode on and off.

F7-1(5) When you drag and drop a virtual control or indicator onto the data screen, the next thing you usually do is resize or move the object around the data screen. Turn the Snap-to-Grid mode on in order to align the virtual control or indicator to the nearest intersection of grid lines. The **Snap to Grid** button is a toggle button, clicking the button toggles the snap-to-grid mode on and off.

F7-1(6) You can combine multiple virtual controls and indicators so you can work with them as though they were a single object. You can resize, move, copy and paste all virtual controls and indicators in a group as a single unit.

After you have grouped virtual controls and indicators, you can still select any single object within the group without un grouping by first selecting the group, and then clicking on the object you want to select.

F7-1(7) µC/Probe allows you to easily align virtual controls and indicators by first selecting the group of objects you want to align and then clicking on one of the following alignment options:

- Left or Right Edges
- Top or Bottom Edges
- Horizontal or Vertical Centers

All the objects are aligned with respect to the first selected item.

F7-1(8) Use the **Units** button to select the grid and ruler's metric system.

F7-1(9) Use the **Show/Hide Grid** button to show and hide the grid lines on the data screen. The snap-to-grid mode still works even if the grid is not visible.

F7-1(10) Use the **Show/Hide Rulers** button to show or hide the ruler. The Show/Hide Rulers button is a toggle button, clicking the button turns the rulers on and off.

F7-1(11) Every time you click the magnifier glass buttons to zoom in and out, µC/Probe zooms in and out by certain zooming factor. Click the **Zoom Factor** button to select a different zooming factor.

F7-1(12) Click the **Full Screen** mode button to hide all the tools except the data screen. The Full Screen button is a toggle button, clicking the button turns the full screen mode on and off.

7-1 µC/PROBE EXAMPLE

In order to demonstrate the previous layout design tools, Figure 7-2 shows an example of a power plant's diagram used as a background to create a control panel with µC/Probe:

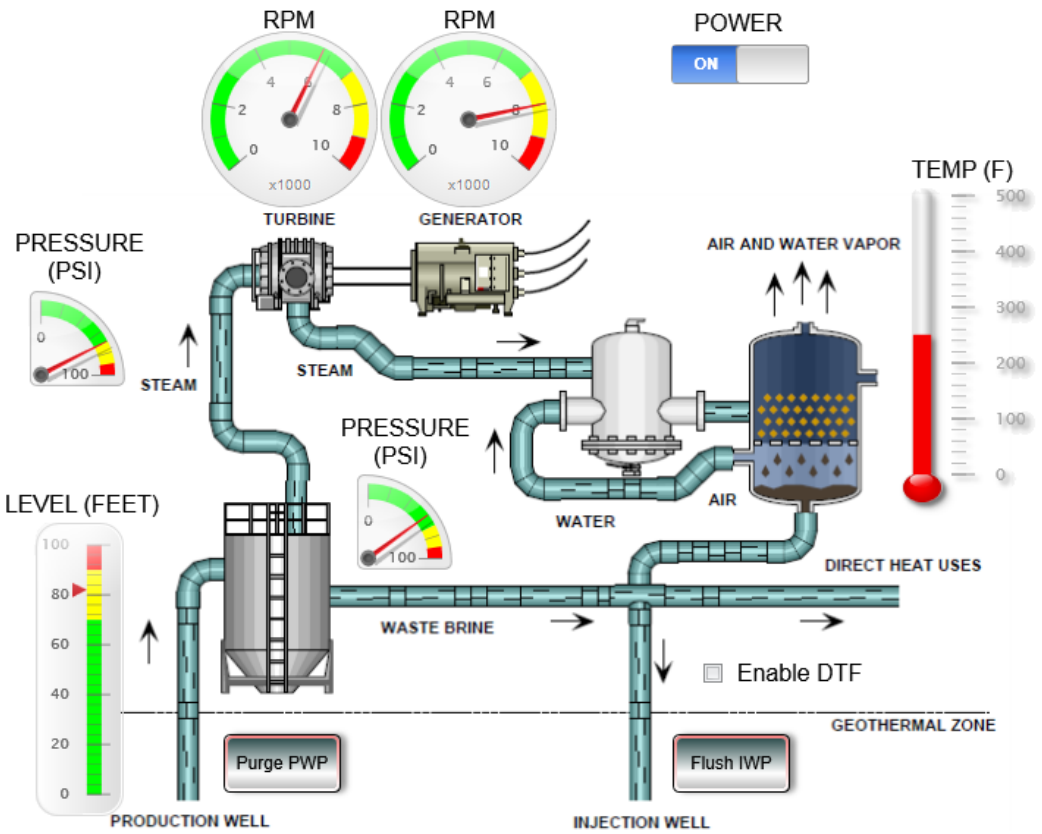


Figure 7-2 **µC/Probe Example of a Power Plant**

Chapter

8

Associating Symbols to Virtual Controls and Indicators

During design-time, use the µC/Probe symbol browser discussed in Chapter 3, “µC/Probe Symbol Browser” on page 15, to search and select the embedded target variables you want to associate to each of the virtual controls and indicators you placed on your data screen.

Once you find the symbol you want to associate, drag and drop the symbol over the virtual control or indicator you want, as shown in Figure 8-1:

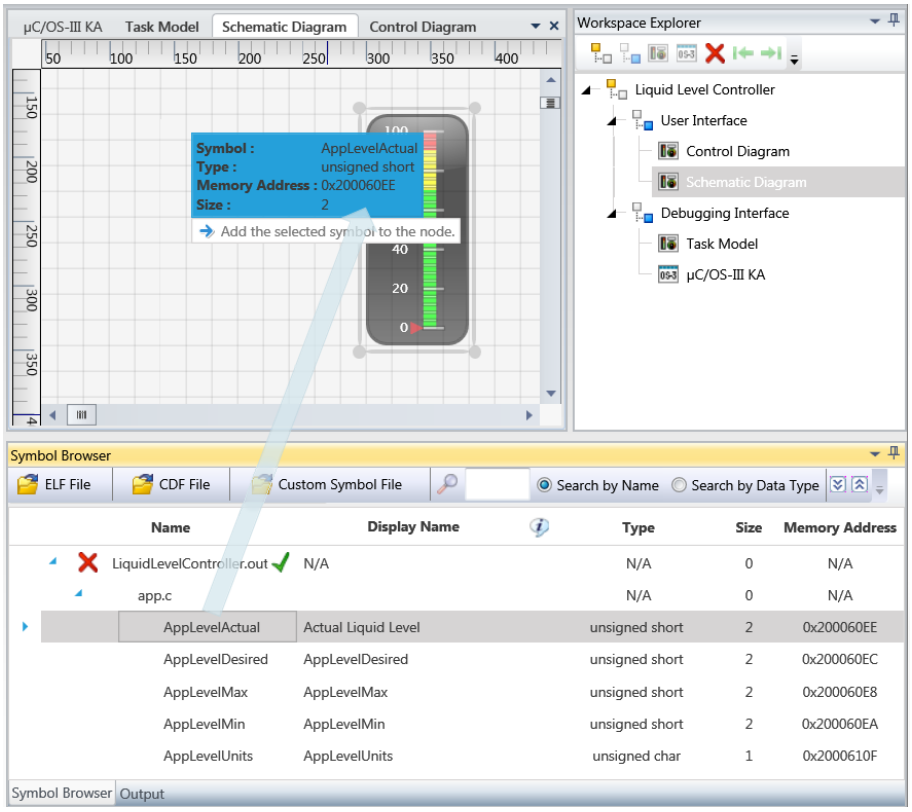


Figure 8-1 **Associating Symbols to Virtual Controls and Indicators**

Repeat the same process for each of the virtual controls and indicators placed on your data screen and µC/Probe will be ready to go into Run Mode unless you want to further configure other optional settings.

In order to configure other optional settings you can use the *symbols manager* by hovering your mouse pointer over the virtual control or indicator and making click on the icon shown in Figure 8-2:

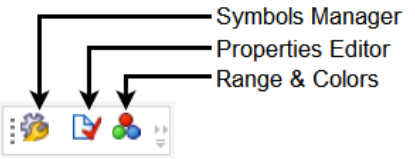


Figure 8-2 **Invoking the Symbols Manager**

Figure 8-3 shows the Symbols Manager:

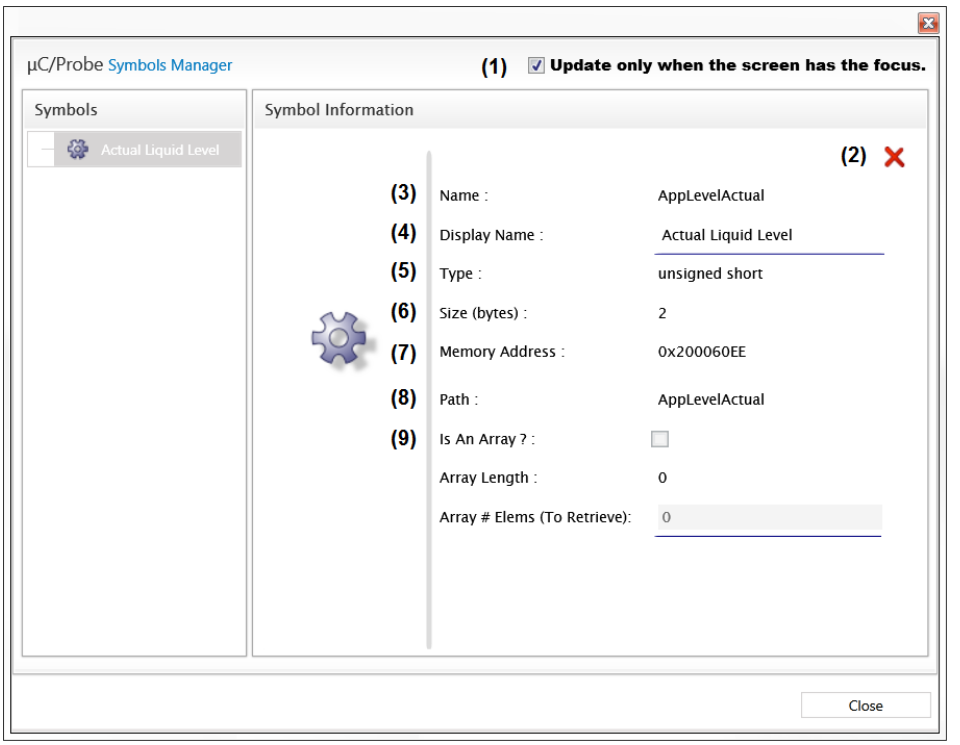


Figure 8-3 **Symbols Manager**

F8-3(1) Tick the checkbox to have µC/Probe update the control's value only when it is visible and in focus.

Untick this checkbox for controls such as charts which you might want to keep updated even when not visible to avoid any gaps in the plot.

F8-3(2) Click the red X to remove the symbol's association from the virtual control or indicator.

F8-3(3) Displays the name of the variable as declared in your C file.

F8-3(4) By default displays the name of the variable as declared in your C file, but this text box allows you to create an alias for display purposes.

F8-3(5) Displays the data type of the variable as declared in your C file.

F8-3(6) Displays the size of the variable in number of bytes.

F8-3(7) Displays the memory location of the variable in the embedded target's memory.

F8-3(8) The variable path displays the full variable name in those cases where the variable you selected is a member of a data structure.

F8-3(9) In case the variable is an array, you can enable indexing of just a certain amount of data.

Chapter

9

Run-Time Mode

9-1 RUN-TIME CHECKLIST

Before setting µC/Probe in Run-Time mode you should verify each of the following items:

Item #	Description	Reference
1	The embedded target has been programmed with an output file (ELF file) with debug information in the DWARF-2, -3 or -4 format or with an XML-based Custom-Symbol File or Chip-Definition File.	µC/Probe Target Manual: Chapter 5, on page 21, Appendix C, on page 55 and Appendix D, on page 61
2	The embedded target is running and connected to the Windows PC through the communication interface of your choice.	µC/Probe Target Manual: Chapter 3, on page 14
3	µC/Probe has been configured with the latest output file (ELF file) that the embedded target is actually running.	µC/Probe User's Manual: Chapter 3, on page 15
4	µC/Probe has been configured with the proper communication interface and settings.	µC/Probe User's Manual: Chapter 4, on page 19
5	µC/Probe contains at least one virtual control or indicator on the data screen.	µC/Probe User's Manual: Chapter 7, on page 36
6	µC/Probe has been configured to associate the virtual control or indicator with one of the embedded target's variables displayed in the symbol browser.	µC/Probe User's Manual: Chapter 8, on page 39

Table 9-1 **Run-Time Mode Checklist**

In order to set µC/Probe in Run-Time mode, click on the **run** button indicated in Figure 9-1:

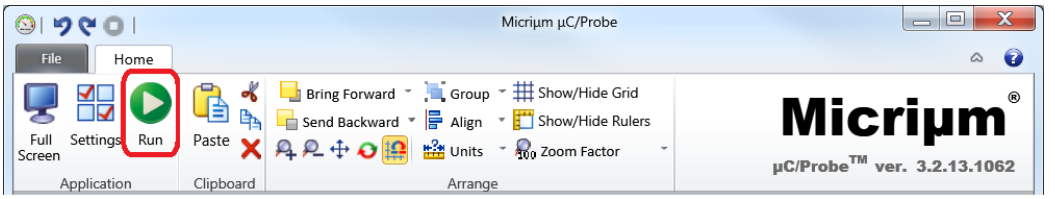


Figure 9-1 **Setting µC/Probe in Run-Time mode**

µC/Probe should start updating your data screens immediately and the application displays all kinds of status information in the status bar at the bottom of the µC/Probe window as shown in Figure 9-2:

Running
Cortex-M3 r1p1, Little endian.
J-Link (SWD) - 12000 kHz
Symbols Configured: 25
Tx: 1,639 requests @ 8.22 KB/sec - Rx: 1,639 responses @ 8.22 KB/sec

Figure 9-2 **µC/Probe Status Bar**

9-2 RUNNING µC/PROBE AND YOUR DEBUGGING SOFTWARE AT THE SAME TIME

Your debugging software for embedded applications usually comes integrated with your IDE and at a minimum, allows you to step through the code, set breakpoints, display register and memory windows, display call stack information, and monitor variables and expressions. Examples of debugging software include IAR's C-SPY and GNU's GDB.

You can also use µC/Probe to extend the capabilities of your debugging software by running both at the same time. µC/Probe allows you to have instant control over your global variables in a real-time and non-intrusive way. From you debugger software, you can set breakpoints at locations of particular interest in the application being debugged and µC/Probe will stop updating the virtual controls and indicators at the same time.

This feature is accomplished by sharing the connection between the Windows PC and the Embedded Target being debugged. Whether the debugger of your choice is IAR's C-SPY, GNU's GDB or any other debugging software that supports J-Link, Figure 9-3 illustrates and example of running µC/Probe and the debugger of your choice at the same time:

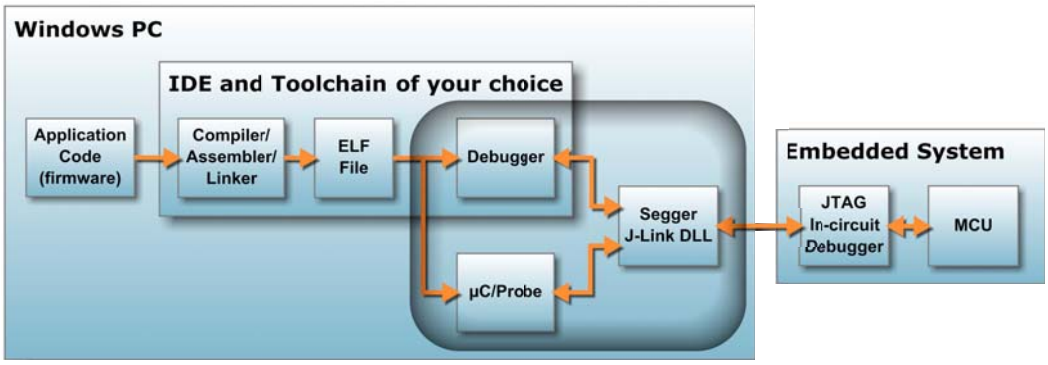


Figure 9-3 **Running µC/Probe and your debugging software at the same time**

Notice how µC/Probe and the Debugger Software not only share the same ELF file but also the same logic and physical interface through the Segger's J-Link DLL and JTAG in-circuit debugger respectively.

9-3 IAR SYSTEMS C-SPY PLUGIN FOR µC/PROBE

µC/Probe is tightly integrated with IAR Embedded Workbench® thanks to a TCP/IP bridge between C-SPY® and µC/Probe. This bridge gives µC/Probe access to not only its native supported platforms but also all the devices and processor architectures supported by IAR Systems without the need to write any target resident code in the form of communication routines, because C-SPY® handles all communication needed as illustrated in the following Figure 9-4:

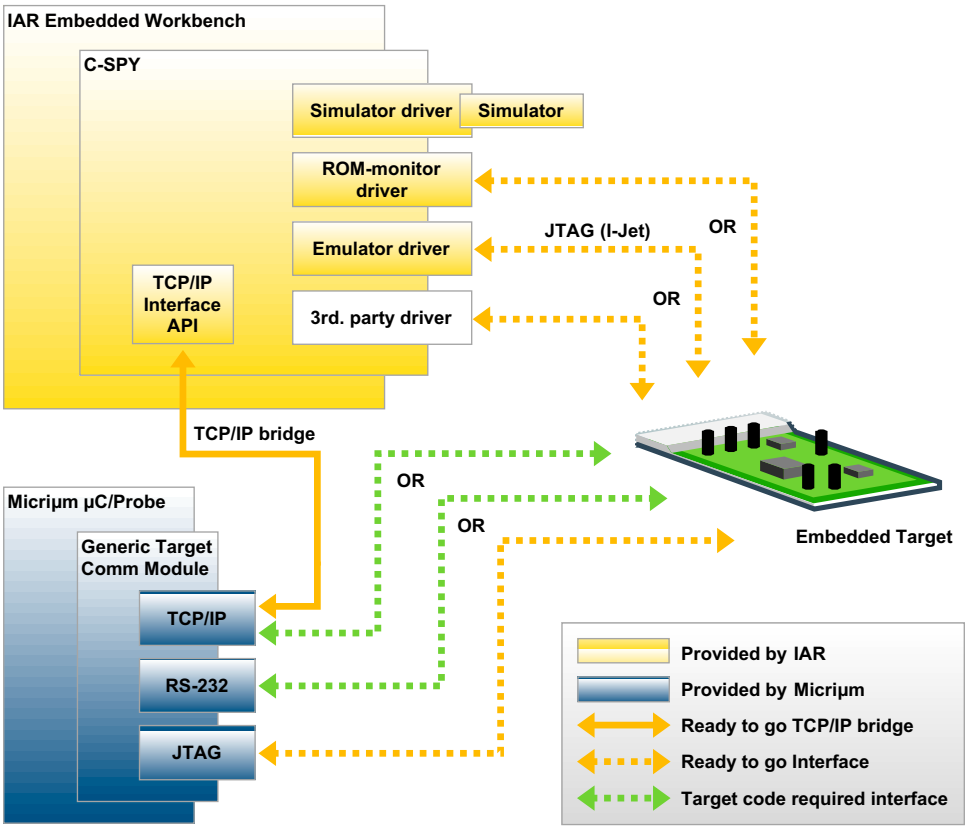


Figure 9-4 **Communication Interfaces Supported by IAR Systems C-SPY and µC/Probe**

In regards to the communication interface options illustrated in Figure 9-4, some of them, such as the ones based on JTAG are ready-to-go assuming your embedded target has either an in-circuit debugger or an external JTAG probe (i.e. IAR's I-Jet). Other communication interfaces require some resident code running in the embedded target, which is available by Micriµm for most platforms.

9-3-1 CONFIGURING THE TCP/IP BRIDGE BETWEEN IAR C-SPY AND µC/PROBE

The TCP/IP bridge between C-SPY® and µC/Probe is built in the form of a plugin module delivered with the Embedded Workbench product installation.

In order to configure Embedded Workbench to load the plugin, you open your project's debugger options and select the plugin from the list of available plugins as shown in Figure 9-5:

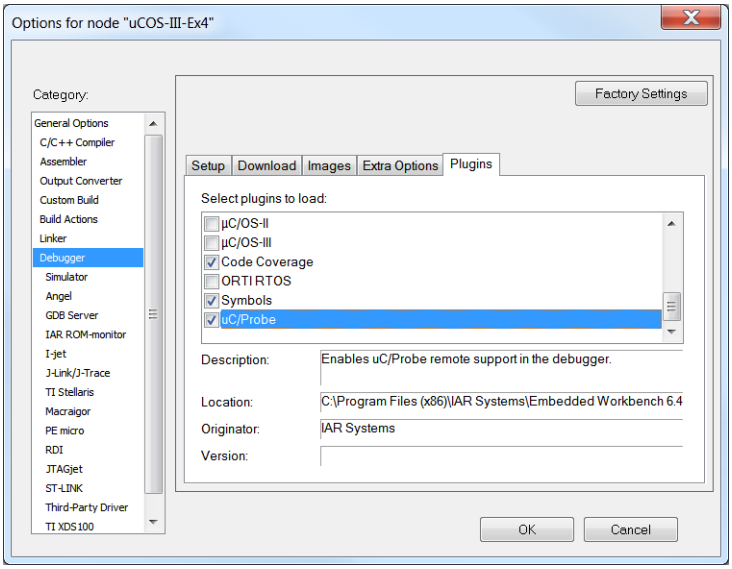


Figure 9-5 **Configuring the IAR Systems Embedded Workbench**

The plugin module gets loaded during a debug session and opens a TCP/IP socket on localhost to listen for µC/Probe requests.

Depending on your network security settings, the first time you launch a debug session you may be asked to allow Embedded Workbench to open a TCP/IP connection.

At the same time, µC/Probe needs to be configured to connect through its TCP/IP interface to localhost on port 9930 as described in section 4-2-3 “TCP/IP” on page 25.

Appendix

A

Configuring Virtual Controls and Indicators

Once you drag and drop one of the virtual controls or indicators onto the data screen and associate it with one of the embedded target's symbols from the symbol browser, you can access the properties tool bar by moving the mouse over the virtual control or indicator.

The tool bar shown in Figure A-1 appears for you to select between one of the three configuration categories:

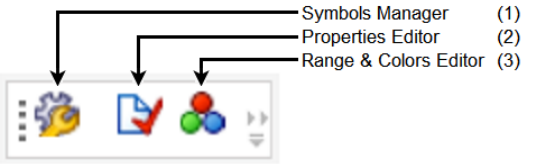


Figure A-1 **Virtual Controls and Indicators Toolbar**

FA-1(1) The Symbols Manager is common for all virtual controls and indicators, see Chapter 8, "Associating Symbols to Virtual Controls and Indicators" on page 39 for more information on the Symbols Manager.

FA-1(2) The Properties Editor is similar among most of the virtual controls and indicators and the next sections will describe how to use the Properties Editor for just a few of the most representative virtual controls and indicators.

FA-1(3) The Range and Colors Editor is only available to those virtual indicators that feature a multi-color scale. The next sections will describe how to use the Range and Colors Editor for a few of the most representative virtual indicators.

A-1 VIRTUAL INDICATORS

A-1-1 FORMATTING PROPERTIES EDITOR

The virtual indicators formatting category applies to linear gauges, half donuts, cylinders, numeric indicators, thermometers, graphs and any virtual indicator capable of showing the symbol's value in a graphical or text format. Figure A-2 shows the formatting category of a linear gauge:

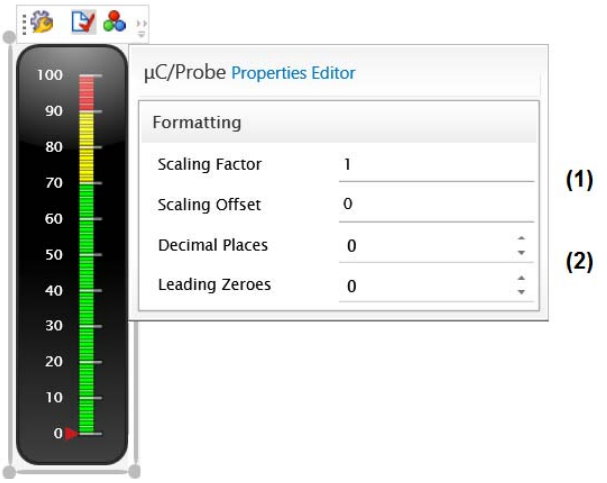


Figure A-2 **Formatting Properties Editor**

FA-2(1) In case you need to convert the value to Engineering Units (EU) before displaying in the virtual indicator, you can use the scaling factor and offset to specify the parameters of a linear conversion function. For example, if the embedded target's symbol you need to display is a 4-20mA value, you can implement the standard linear equation y = mx + b where m is the scaling factor, x is the 4-20mA value, b is the offset and y is the resulting Engineering Units (EU) value to display.

FA-2(2) During run-time mode, whenever you move the mouse pointer over a virtual indicator such as a linear gauge, a small yellow box will appear with the symbol's value formatted with the number of decimal places and leading zeros of your choice.

A-1-2 RANGE AND COLORS EDITOR

The Range and Colors Editor applies to linear gauges, half donuts, cylinders and any virtual indicator capable of displaying the symbol's value in a graphical format along a multi-color scale. Figure A-3 shows the Range and Colors Editor for a linear gauge:

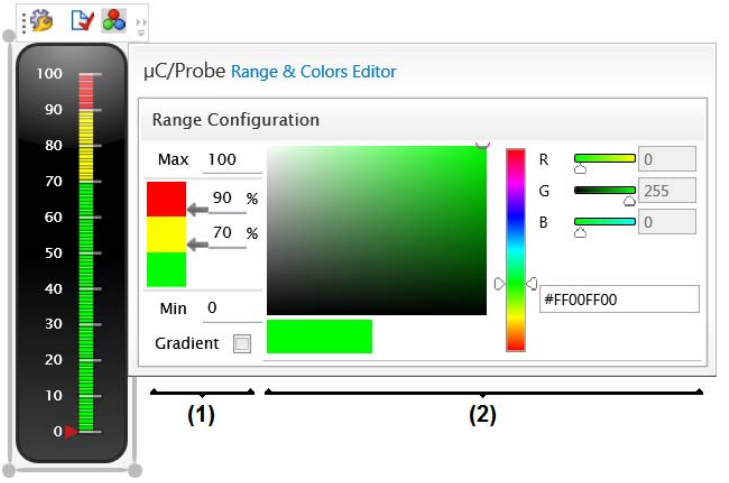


Figure A-3 **Range and Colors Editor**

FA-3(1) Start by setting the **Min** and **Max** limits, then click on the text boxes next to the percentage signs and enter the thresholds in terms of percentage.

FA-3(2) Each time you click on one of the text boxes to set the percentages, the color picker allows you to choose the color for that gauge band. You can enter the color you want in hex format or by selecting a color from the vertical slider and then fine tuning with the palette.

A-1-3 NUMERIC INDICATOR PROPERTIES EDITOR

The Numeric Indicator category from the Properties Editor only applies to numeric indicators. Figure A-4 shows the numeric indicator's properties. Font styles, alignment and the thousand separator, they all apply to the number 0 shown in white:

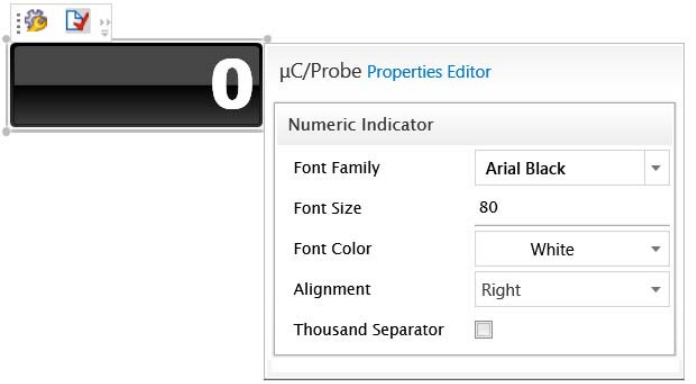


Figure A-4 **Numeric Indicator Properties Editor**

A-1-4 BITMAP ANIMATION PROPERTIES EDITOR

The bitmap animations are part of the toolbox's miscellaneous family. They are one of the most powerful virtual indicators because you have the freedom to customize it however you want by providing your own images.

Imagine you want to display the state of a valve to be either open or closed in a graphical way by using the bitmap images shown in Figure A-5 and an embedded target's application variable named `AppValveOutPct` that stores the state of the outflow valve (0%:open and 100%:closed).

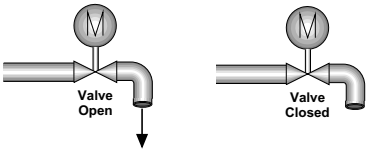


Figure A-5 **Bitmaps to Animate**

Figure A-6 shows the properties editor for the bitmap animation:

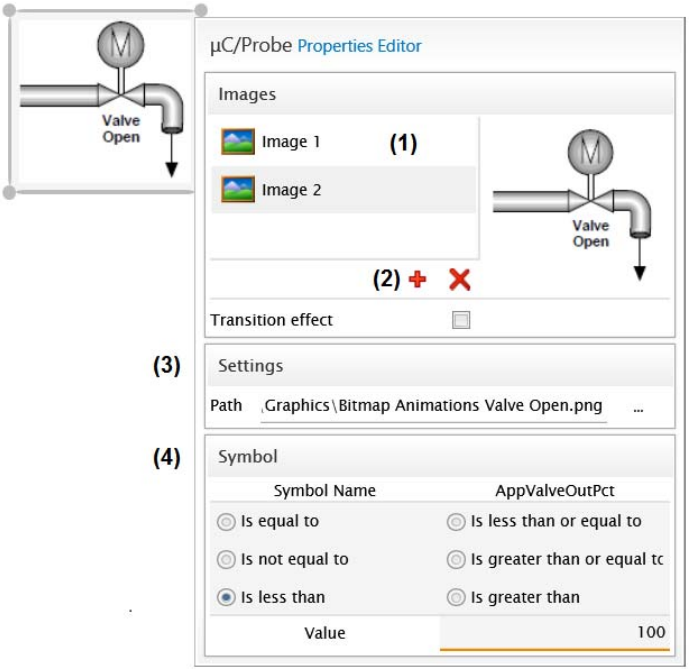


Figure A-6 **Bitmap Animation Properties Editor**

FA-6(1) The list of images is initialized with two images. Start by selecting the image you want to work with. A preview of the image is shown on the right side.

FA-6(2) You can add or delete more images into the list by making click on the red **+** or **x** buttons respectively.

Select the transition effect check box if you want to add a fade-in and fade-out effect between image transitions.

FA-6(3) Specify the bitmap file path or browse to it.

FA-6(4) Specify the rules. In this case, if `AppValveOutPct < 100` then the valve is open, and if `AppValveOutPct = 100` then the valve is all the way closed.

A-2 VIRTUAL CONTROLS

A-2-1 SLIDER CONTROL PROPERTIES EDITOR

The slider control is one of the few writable controls. It allows you to gradually modify an adjustable embedded target symbol's value. The user gets to select from a range of values by moving a value indicator up and down a track. For example, you typically create a volume control by using a slider control.

Figure A-7 shows the slider control properties editor:

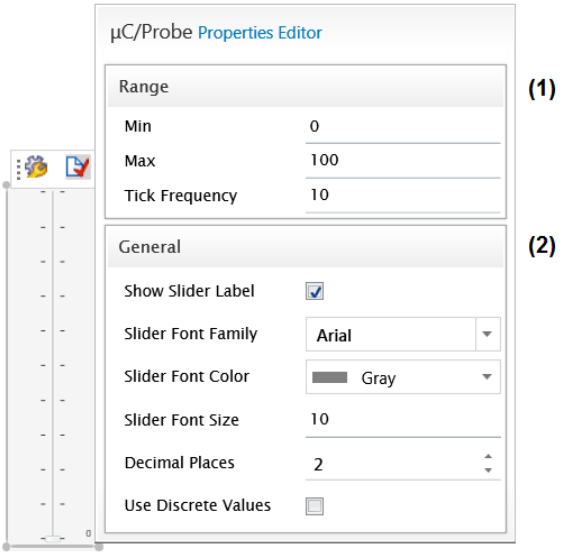


Figure A-7 **Slider Control Properties Editor**

FA-7(1) The slider control has a minimum, a maximum, and an increment value. The **Tick Frequency** not only determines the increment value but also the number of tick marks along the track.

FA-7(2) The **General** category includes the formatting properties that affect the slider's tick labels. Select the check box **Use Discrete Values** if you want the slider to adjust the associated symbol by making discrete increments.

A-2-2 CUSTOM SLIDER PROPERTIES EDITOR

The custom slider is similar to the one from section A-2-1, except that it also allows you to include two images to the left and right side of the slider's track as shown in Figure A-8:

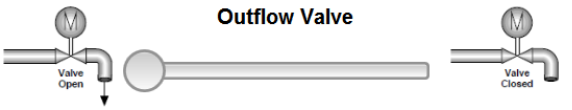


Figure A-8 **Custom Slider Example**

Imagine you want to control the state of a valve to be either open or closed by modifying an embedded target's application variable named `AppValveOutPct` that stores the state of the outflow valve (0%:open and 100%:closed). Figure A-9 shows the custom slider properties editor for such example:.

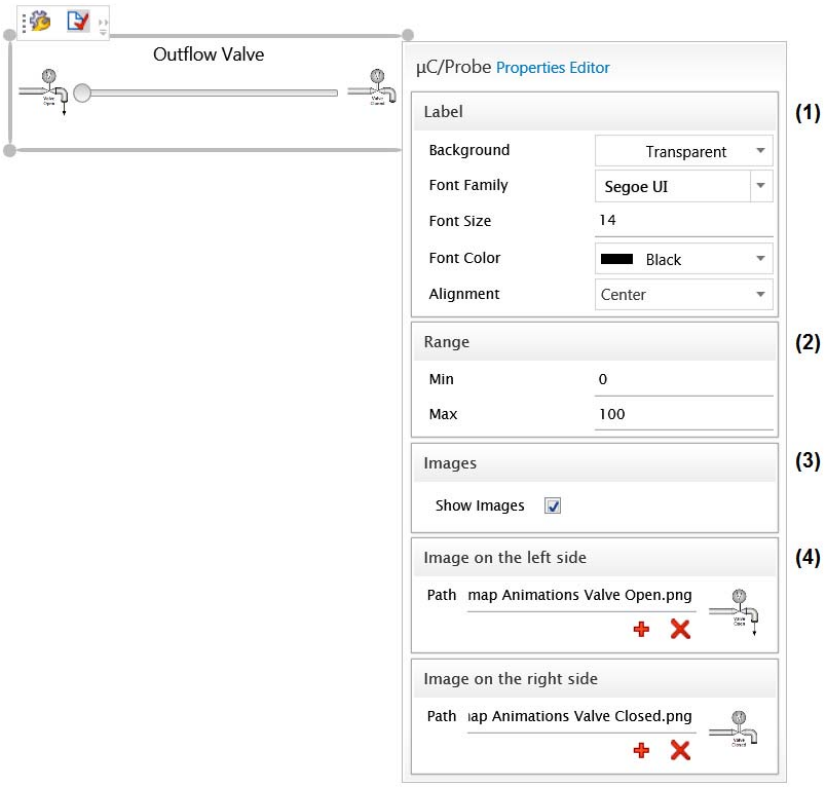


Figure A-9 **Custom Slider Properties Editor**

FA-9(1) The label in this example is *Outflow Valve*. The label properties include the background color, font styles and alignment.

FA-9(2) The custom slider control has a minimum and maximum range.

FA-9(3) You can turn on and off the images.

FA-9(4) Specify the path or browse to the bitmap you want to be placed on the left and right sides of the track.

A-2-3 CUSTOM SWITCH PROPERTIES EDITOR

The custom switch control is a two state button. You can modify the value of an embedded target's symbol by specifying the values you want to write when the button is switched between the **On** and **Off** states as shown in Figure A-10:

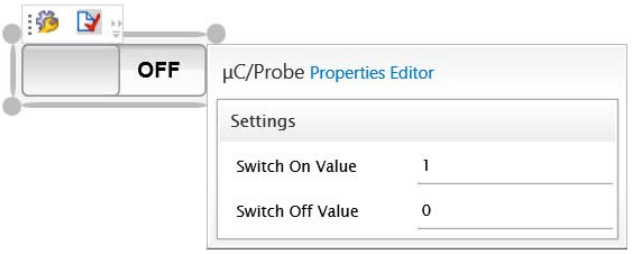


Figure A-10 **Custom Switch Properties Editor**

A-2-4 CHECKBOX PROPERTIES EDITOR

The checkbox control is similar to the custom switch but it also allows you to specify a label to display when the checkbox is selected and not selected. In the example shown in Figure A-11 such labels are set to Enabled during the On state and Disabled during the Off state.

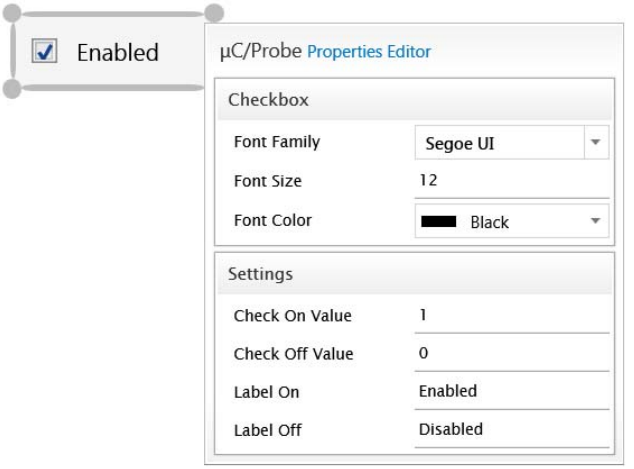


Figure A-11 **Checkbox Properties Editor**

A-2-5 PUSH BUTTON PROPERTIES EDITOR

The push button control is a momentary switch that switches between the states **On** while held down and **Off** when released. The properties window is shown in Figure A-12.

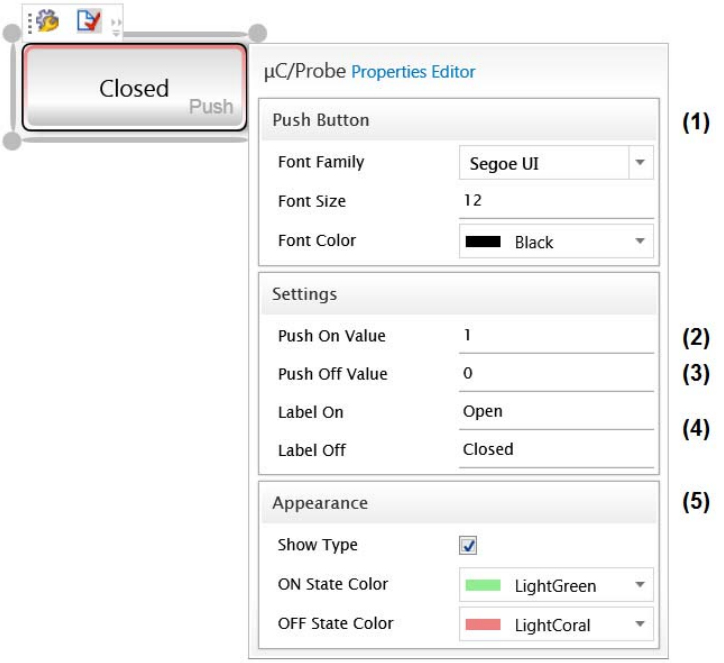


Figure A-12 **Push Button Properties Editor**

FA-12(1) You can specify the labels to display during the **On** and **Off** states. These properties affect the label's font family, font size and font color properties.

FA-12(2) During run-time, when the user holds down the push button, µC/Probe writes the **Push On Value** one single time to the embedded target.

FA-12(3) When the user releases the button, µC/Probe writes the **Push Off Value** one single time to the embedded target.

FA-12(4) Here you specify the labels you want to display during the **On** and **Off** states.

FA-12(5) You can also specify the border colors you want to display during the **On** and **Off** states and whether or not you want to show the **Push** label on the corner.

A-2-6 TOGGLE BUTTON PROPERTIES EDITOR

The toggle button control is a button that switches between the states **On** and **Off** when clicked. The properties window for a relay's toggle button is shown in Figure A-13.

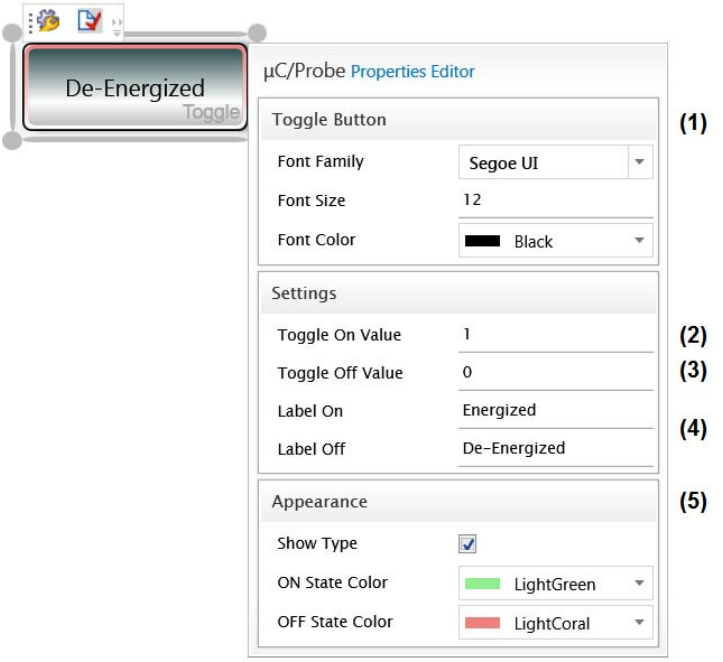


Figure A-13 **Toggle Button Properties Editor**

FA-13(1) You can specify the labels to display during the **On** and **Off** states. These properties affect the label's font family, font size and font color properties.

FA-13(2) During run-time, when the user clicks and releases the button, µC/Probe writes the **Toggle On Value** one single time to the embedded target.

FA-13(3) When the user clicks and releases the button again, µC/Probe writes the **Toggle Off Value** one single time to the embedded target.

FA-13(4) Here you specify the labels you want to display during the **On** and **Off** states.

FA-13(5) You can specify the border colors you want to display during the **On** and **Off** states and whether or not you want to show the **Toggle** label on the corner.

A-2-7 REPEAT BUTTON PROPERTIES EDITOR

The repeat button control is a button that switches between the states **On** while held down and **Off** when released. The properties window is shown in Figure A-14.

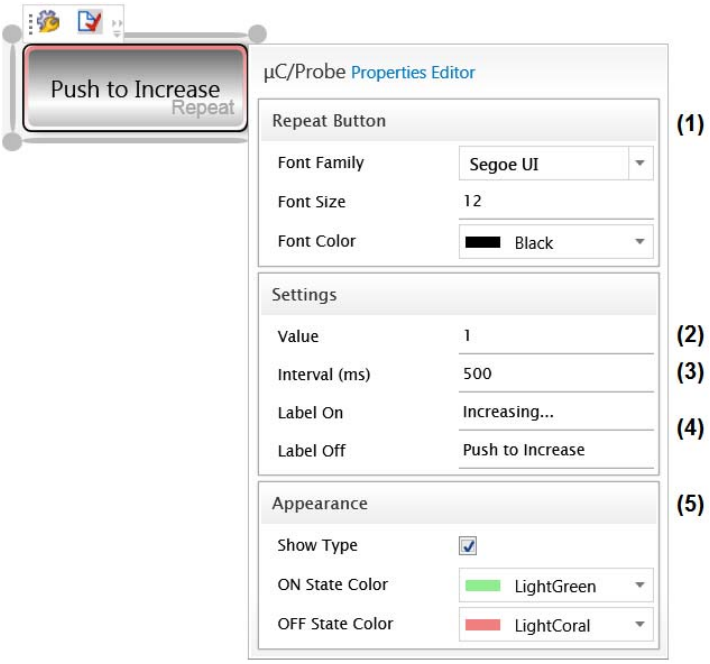


Figure A-14 **Repeat Button Properties Editor**

FA-14(1) You can specify the labels to display during the **On** and **Off** states. These properties affect the label's font family, font size and font color properties.

FA-14(2) During run-time, when the user holds down the repeat button, µC/Probe writes the **Value** multiple times to the embedded target until the button is released.

FA-14(3) The value gets written to the embedded target multiple times at the specified interval in milliseconds.

FA-14(4) Here you specify the labels you want to display during the **On** and **Off** states.

FA-14(5) You can specify the border colors you want to display during the **On** and **Off** states and whether or not you want to show the **Repeat** label on the corner.

A-2-8 BIT CONTROL PROPERTIES EDITOR

The bit control is part of the writable controls category in µC/Probe's toolbox. It allows you to read and write to a symbol by either toggling its bits on and off or entering the value in either decimal or hexadecimal format.

This control is perfect to represent peripheral I/O registers and the properties window is shown in Figure A-15:

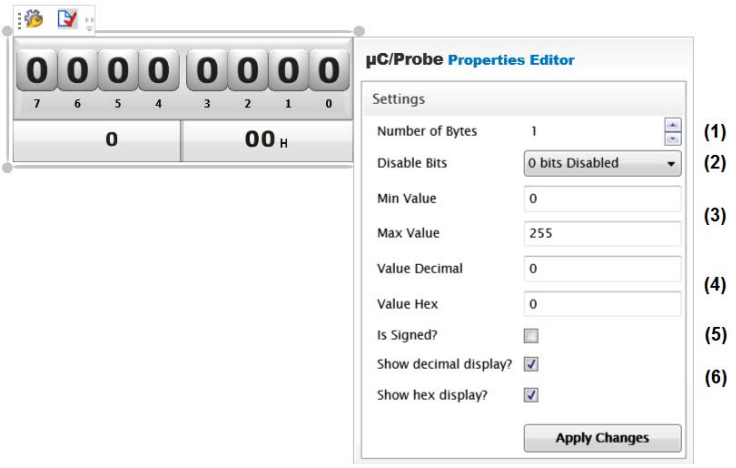


Figure A-15 **Bit Control Properties Editor**

FA-15(1) You can specify the number of bytes in case you want to override the original data size.

FA-15(2) You can also disable bits, in those cases where some of the bits are reserved which is typically the case with peripheral I/O registers. This drop-down gives you a list of all the bits along with a checkbox to enable and disable each bit.

FA-15(3) You can specify the minimum and maximum value range allowed.

FA-15(4) You can specify an initial value in either decimal or hexadecimal format.

FA-15(5) You can override the data type and specify whether it is signed or unsigned.

FA-15(6) Finally, you can hide the decimal and hexadecimal displays in case you want to show the bit buttons only.

A-3 CHARTS

µC/Probe supports two types of charts; timeline charts and scatter x-y charts.

A-3-1 TIMELINE CHARTS

Timeline charts are those whose samples are considered events in time and the horizontal axis represents the time. Figure A-16 shows the three types of timeline charts supported by µC/Probe:

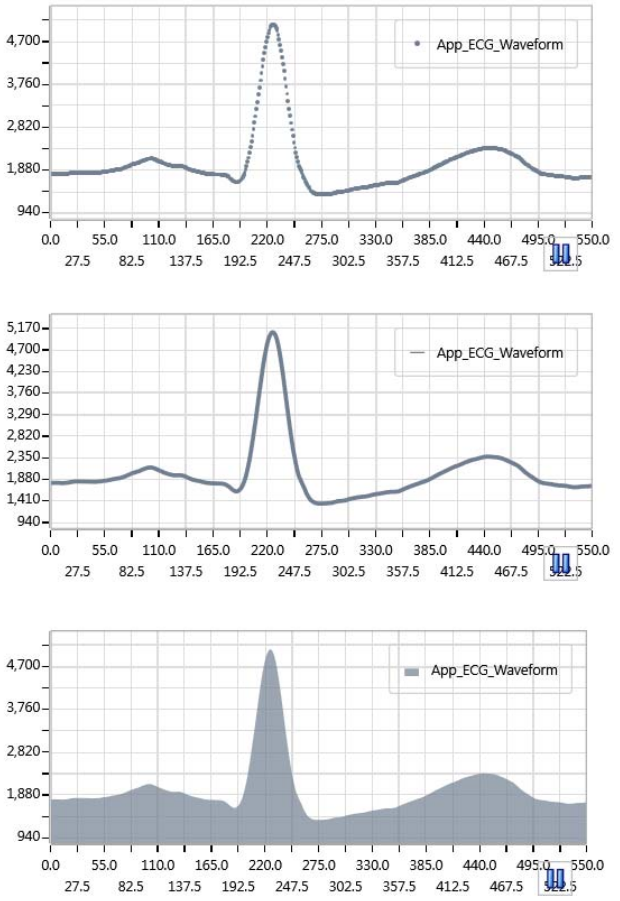


Figure A-16 **Marker, Line and Area Charts**

TIMELINE CHART PROPERTIES EDITOR

All three timeline chart types share the same properties editor as shown in Figure A-17:

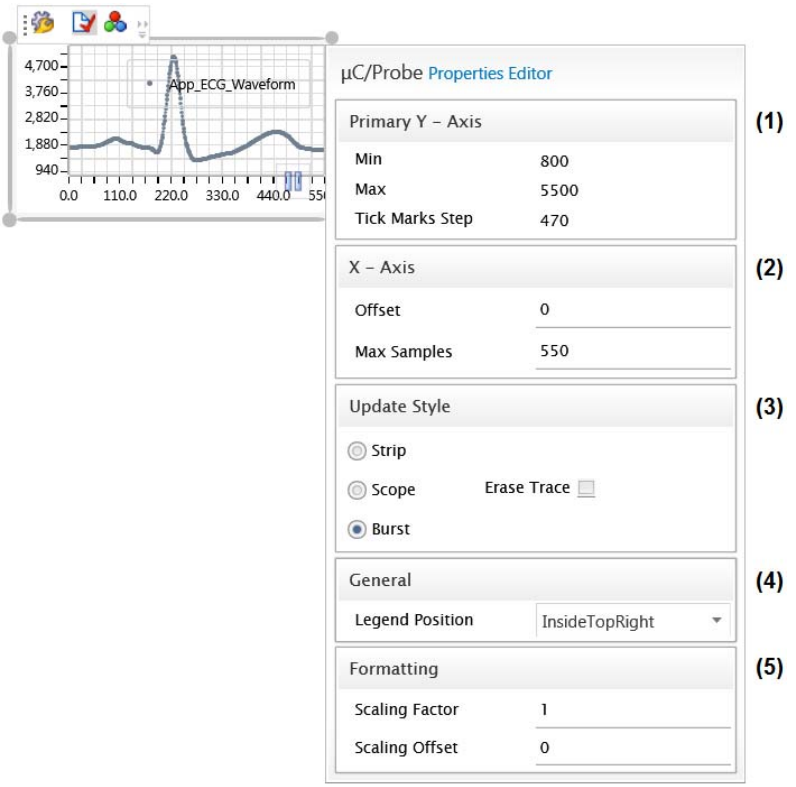


Figure A-17 **Charts Properties Editor**

FA-17(1) In order to change the number at which the vertical axis value starts or ends, enter a different number in the **Min** box or the **Max** box.

In order to change the interval of tick marks and chart grid lines, enter a different number in the **Tick Marks Step** box.

FA-17(2) In order to change the number at which the horizontal axis value starts or ends, enter a different number in the **Offset** box or the **Max Samples** box.

FA-17(3) µC/Probe updates charts in one of three modes:

- **Strip** Mode: similar to a chart recorder device that prints over a paper strip, µC/Probe first plots points from the left to the right side of the chart. From there, any new points are plotted at the rightmost side of the chart by shifting old points to the left.

- **Scope** Mode: similar to an oscilloscope, µC/Probe first plots points from the left to the right side of the chart. Once the plot reaches the right side of the chart, it erases the plot and begins plotting again from the left side of the chart.

- **Burst** Mode: this update mode was made for high performance applications where you want to plot array data quickly by plotting the entire array in one sweep. µC/Probe will not shift any points over the plotting area. Instead, it will erase the plot and will plot the same array again, assuming that the array is being updated by the embedded target.

FA-17(4) µC/Probe supports charts with multiple data series. That means that you can associate multiple symbols from your embedded target into one single chart.

In order to tell which trace represents a symbol in your chart, a color-coded legend with the name of the symbol is displayed over the chart. Select the legend position that better suits your needs.

FA-17(5) In case you need to convert the value points to Engineering Units (EU) before plotting in the chart, you can use the scaling factor and offset to specify the parameters of a linear conversion function. For example, if the embedded target's symbol you need to display is a 4-20mA value, you can implement the standard linear equation y = mx + b where m is the scaling factor, x is the 4-20mA value, b is the offset and y is the resulting Engineering Units (EU) value to be plotted.

TIMELINE CHART SERIES EDITOR

The charts series editor allows you to configure each trace in the plotting area. The series editor is shared by all three types of charts as shown in Figure A-18:

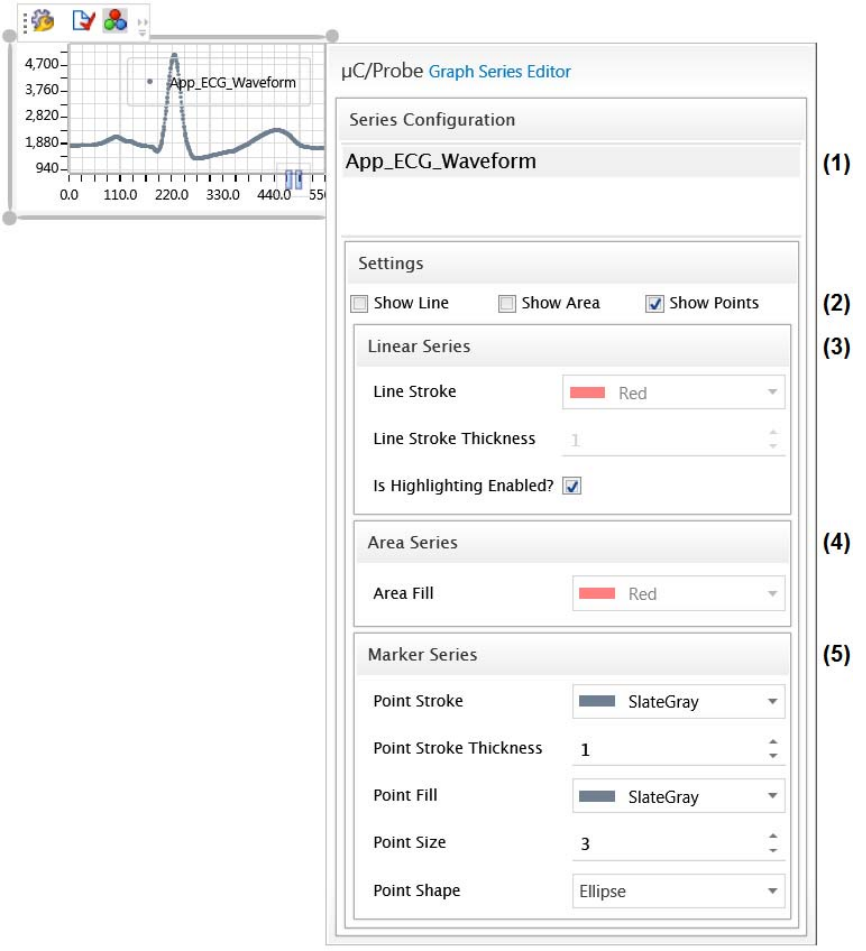


Figure A-18 **Charts Series Editor**

FA-18(1) Select the data series you want to work with. By default, the name of the data series is the same of the symbol as declared in your embedded target C files.

FA-18(2) µC/Probe supports three types of charts: marker, line and area charts. You can mix and match chart types in one single chart by selecting the corresponding check box.

FA-18(3) If the **Show Line** check box is selected, this section allows you to configure the color and thickness of the line.

FA-18(4) If the **Show Area** check box is selected, this section allows you to configure the color of the area.

FA-18(5) If the **Show Points** check box is selected, this section allows you to configure the color, thickness and shape of the points.

A-3-2 SCATTER X-Y CHARTS

Contrary to timeline charts where the horizontal axis is a representation of time, the scatter x-y charts allow you to use two different arrays to specify the x-y coordinates of each data point.

Use a scatter x-y chart if the data you want to plot includes pairs of values and you want to compare data points without regard of time.

The scatter x-y chart supports two modes of operation; burst and plot mode.

SCATTER X-Y CHART IN BURST MODE

Use the *burst mode* to compare sets of values stored in two different arrays.

For example, imagine a blood pressure monitor that stores the diastolic pressure in mmHg for each patient. You can use the scatter x-y chart in burst mode to compare the diastolic pressure among the patients age groups as follows.

Code Listing A-1 shows the two arrays that contain the data points.

```
static  const  CPU_INT08U  AppAgeTbl[20] =
{
    20, 28, 32, 45, 26, 64, 23, 54, 32, 54, 23, 33, 44, 21, 43, 52, 56, 62, 23, 45
};

static  const  CPU_INT08U  AppDiastolicPressureTbl[20] =
{
    64, 67, 72, 76, 65, 100, 64, 94, 80, 90, 70, 73, 81, 62, 87, 89, 91, 98, 68, 87
};
```

Listing A-1 **Two Arrays**

First you drag-and-drop an instance of the scatter x-y chart into a data screen. Then you open the symbols file (i.e. ELF file), search for the two arrays `AppAgeTbl[]` and `AppDiastolicPressure[]`, and drag-and-drop the arrays over the scatter x-y chart.

If you open the properties of the chart you will see something similar to Figure A-19:

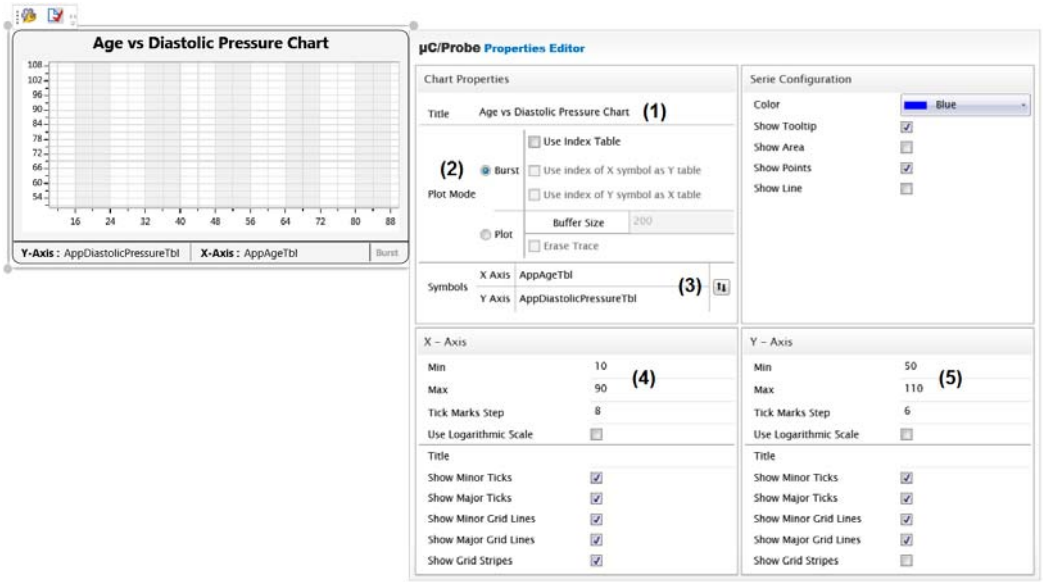


Figure A-19 **Scatter X-Y Chart Properties**

FA-19(1) Enter the title for your scatter x-y chart.

FA-19(2) µC/Probe will automatically set your plot mode to *Burst Mode* in case the axis have been mapped to arrays.

In case you only have one single array and want to use the array indexes as one of the axis, you can enable the checkbox *Use Index Table* and select which axis you want the index to be.

FA-19(3) You can use this button to swap the axis.

FA-19(4) Set the range of your data points over the horizontal axis.

FA-19(5) Set the range of your data points over the vertical axis.

During run-time, µC/Probe and the scatter X-Y chart should plot all the data points at once and will keep refreshing the plot as the values of the arrays change. You should see a scatter chart similar to the one in Figure A-20:

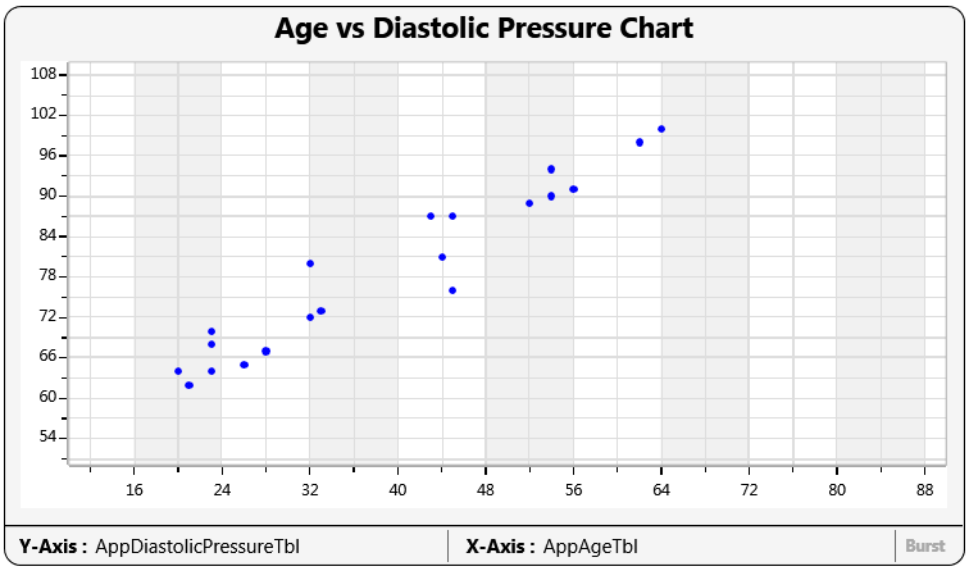


Figure A-20 **Scatter X-Y Chart in Burst Mode**

SCATTER X-Y CHART IN PLOT MODE

For the *plot mode* of the X-Y scatter chart, imagine for instance you want to monitor in real-time the power curve of a wind turbine. Your embedded application does not buffer any data and instead, it only keeps the current wind speed (meters/sec) and current power (watts) stored in the global variables `AppWindSpd` and `AppPwr` respectively.

Similar to the previous example, you drag-and-drop an instance of the scatter x-y chart into a data screen. Then, you open the symbols file (i.e. ELF file), search for the two variables `AppWindSpd` and `AppPwr`, and drag-and-drop the them over the scatter x-y chart.

If you open the properties of the chart you will see something similar to Figure A-21:

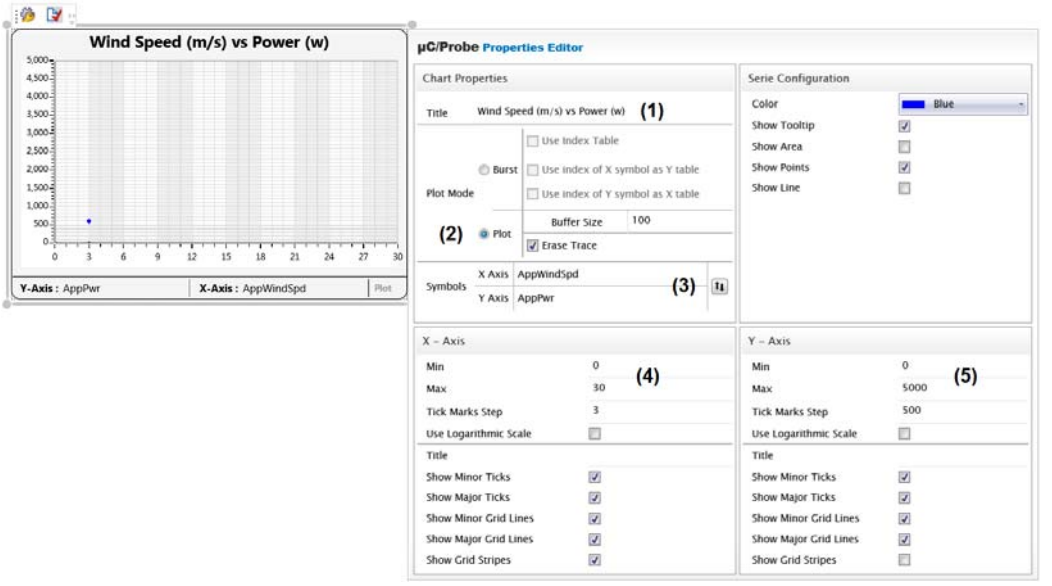


Figure A-21 **Scatter X-Y Chart Properties**

FA-21(1) Enter the title for your scatter x-y chart.

FA-21(2) µC/Probe will automatically set your chart to *Plot Mode* in case the axis have been mapped to a pair of non-array variables.

In this case, you need to specify the size of the host-side buffer that will hold the pair of values and whether or not you want the entire plot to be erased once the buffer is full.

FA-21(3) You can use this button to swap the axis.

FA-21(4) Set the range of your data points over the horizontal axis.

FA-21(5) Set the range of your data points over the vertical axis.

During run-time, the scatter chart should start to fill the plot area with one data point at a time and at the coordinate (`AppWindSpd`, `AppPwr`). Depending on the data change rate and the data collection rate, eventually the chart should look similar to the one in Figure A-22:

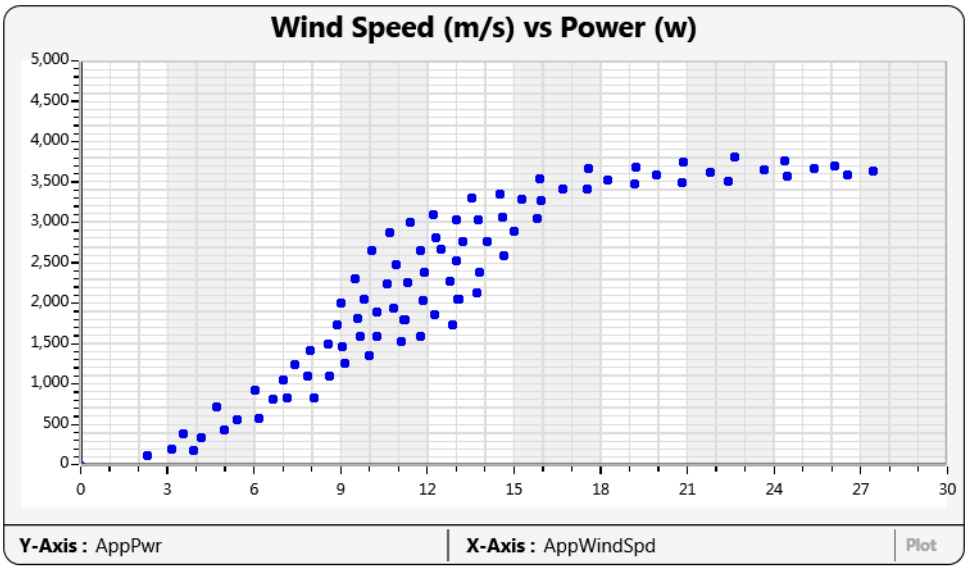


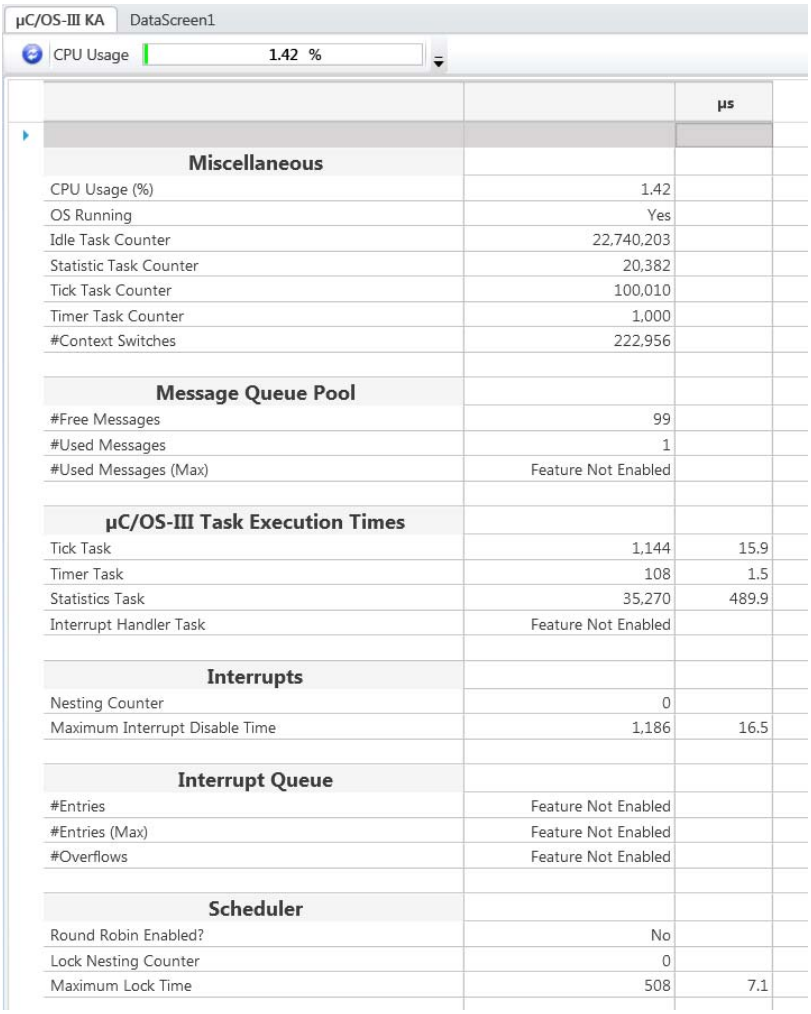
Figure A-22 **Scatter X-Y Chart in Plot Mode**

Appendix B

Kernel Awareness Screen

µC/Probe allows you to add pre-configured data screens that display some of the most popular Micriμm's software modules.

Figure B-1 and Figure B-2 show examples of the Kernel Awareness Screens for µC/OS-III. The Screens display µC/OS-III's internal data structures in a convenient series of windows. This provides you with information about each of the active tasks in the embedded target application among other kernel objects such as semaphores, event flags, mutexes, etc.



		µs
Miscellaneous		
CPU Usage (%)	1.42	
OS Running	Yes	
Idle Task Counter	22,740,203	
Statistic Task Counter	20,382	
Tick Task Counter	100,010	
Timer Task Counter	1,000	
#Context Switches	222,956	
Message Queue Pool		
#Free Messages	99	
#Used Messages	1	
#Used Messages (Max)	Feature Not Enabled	
µC/OS-III Task Execution Times		
Tick Task	1,144	15.9
Timer Task	108	1.5
Statistics Task	35,270	489.9
Interrupt Handler Task	Feature Not Enabled	
Interrupts		
Nesting Counter	0	
Maximum Interrupt Disable Time	1,186	16.5
Interrupt Queue		
#Entries	Feature Not Enabled	
#Entries (Max)	Feature Not Enabled	
#Overflows	Feature Not Enabled	
Scheduler		
Round Robin Enabled?	No	
Lock Nesting Counter	0	
Maximum Lock Time	508	7.1

Figure B-1 **Kernel Awareness Screen: Miscellaneous**

Figure B-2 shows the information displayed for each task. The columns can be sorted and the tri-color bar graphs highlight those tasks reaching their maximum stack space, which is a typical bug when developing embedded systems:

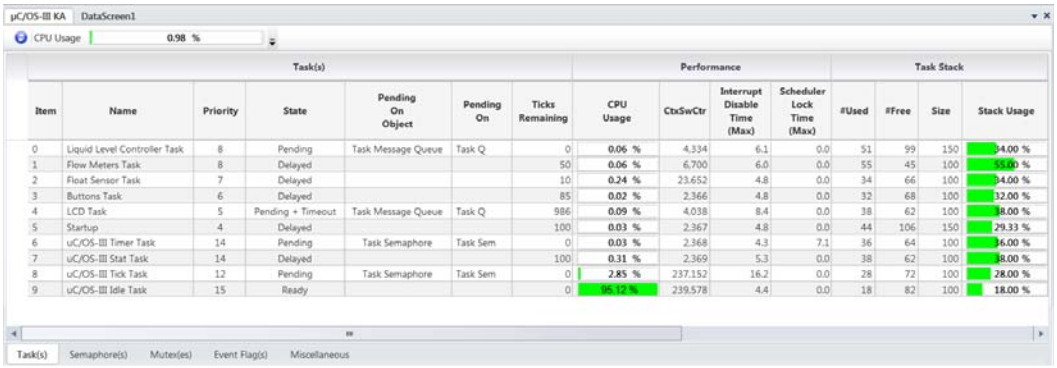

µC/OS-III KA
DataScreen1
CPU Usage
0.98 %
Task(s)
Performance
Task Stack
Item
Name
Priority
State
Pending On Object
Pending On
Ticks Remaining
CPU Usage
CtxSwCtr
Interrupt Disable Time (Max)
Scheduler Lock Time (Max)
#Used
#Free
Size
Stack Usage
0 Liquid Level Controller Task 8 Pending Task Message Queue Task Q 0 0.06 % 4,334 6.1 0.0 51 99 150 34.00 %
1 Flow Meters Task 8 Delayed 50 0.06 % 6,700 6.0 0.0 55 45 100 55.00 %
2 Float Sensor Task 7 Delayed 10 0.24 % 23,652 4.8 0.0 34 66 100 34.00 %
3 Buttons Task 6 Delayed 85 0.02 % 2,366 4.8 0.0 32 68 100 32.00 %
4 LCD Task 5 Pending + Timeout Task Message Queue Task Q 986 0.09 % 4,038 8.4 0.0 38 62 100 38.00 %
5 Startup 4 Delayed 100 0.03 % 2,367 4.8 0.0 44 106 150 29.33 %
6 uC/OS-III Timer Task 14 Pending Task Semaphore Task Sem 0 0.03 % 2,368 4.3 7.1 36 64 100 36.00 %
7 uC/OS-III Stat Task 14 Delayed 100 0.31 % 2,369 5.3 0.0 38 62 100 38.00 %
8 uC/OS-III Tick Task 12 Pending Task Semaphore Task Sem 0 2.85 % 237,152 16.2 0.0 28 72 100 28.00 %
9 uC/OS-III Idle Task 15 Ready 0 95.12 % 239,578 4.4 0.0 18 82 100 18.00 %
Task(s)
Semaphore(s)
Mutex(es)
Event Flag(s)
Miscellaneous

Figure B-2 **Kernel Awareness Screen: Task List**

Appendix

C

Terminal Window Control

µC/Probe provides an option to enable debug traces to output the embedded target's activity via any of the communication interfaces supported by µC/Probe. A trace message is displayed in a terminal window control in µC/Probe, by calling a function `ProbeTermTrcPrint()` from your embedded application as illustrated in Figure C-1. Additionally, you can prefix the messages with special tags that µC/Probe will replace with icons that you get to choose.

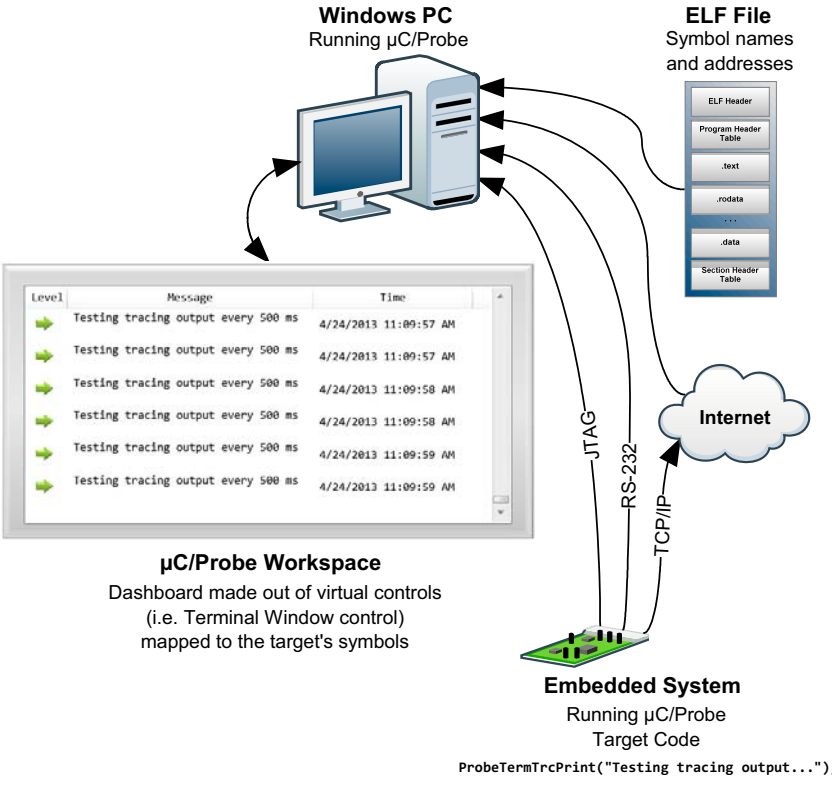


Figure C-1 **Terminal Window Control - Trace Interface**

At the same time, µC/Probe provides the option to enable a command-line interface to the embedded target. A command-line interface allows the user to issue a command to the target from a terminal window control in µC/Probe. Examples of command lines are `ipconfig`, `dir` or whatever command the programmer wants to implement in the embedded target.

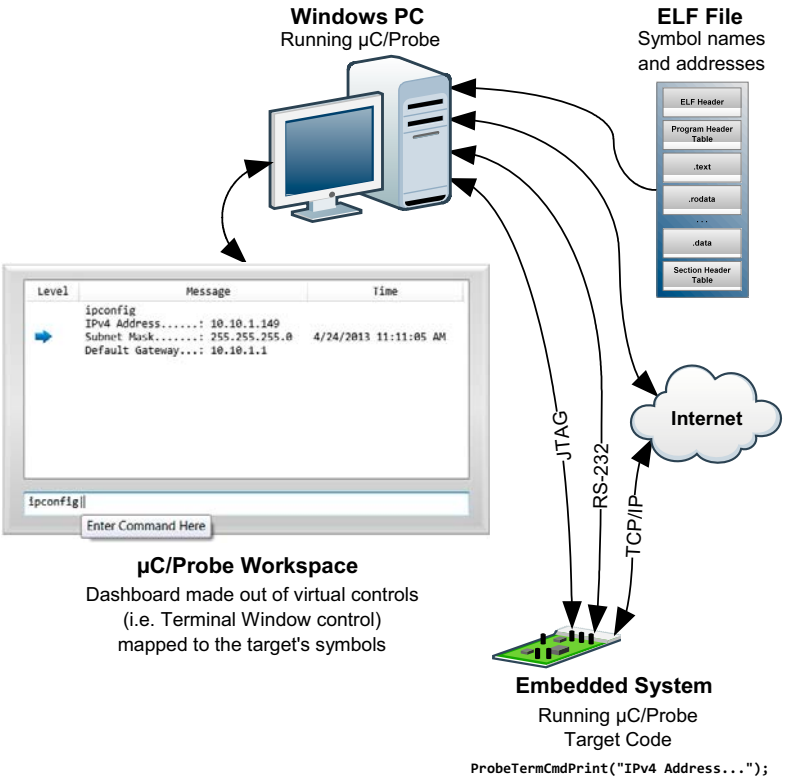


Figure C-2 **Terminal Window Control - Command Line Interface**

This appendix will introduce you to the debug trace and command-line tools available in µC/Probe. It will show you how to configure the control in µC/Probe. More information on the Terminal Window control such as how to include it in your embedded target code and make use of it are in the target version of the µC/Probe manual.

C-1 TERMINAL WINDOW CONTROL CONFIGURATION

The terminal window control is found in the miscellaneous category of µC/Probe's toolbox. Once you drag and drop an instance of this control onto the data screen, you do not need to associate it with any of the embedded target's symbols from the symbol browser, as this is done automatically, assuming you have included the required target code to support terminal window as described in the target version of the µC/Probe manual.

You can access the properties tool bar by moving the mouse over the terminal window control. The tool bar shown in Figure C-3 appears for you to select between one of the two configuration categories:

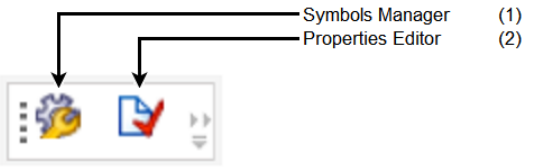


Figure C-3 **Terminal Window Control Toolbar**

FC-3(1) The Symbols Manager is common for all virtual controls and indicators, see Chapter 8, "Associating Symbols to Virtual Controls and Indicators" on page 37 for more information on the Symbols Manager.

FC-3(2) The Properties Editor is similar among most of the virtual controls and indicators and the next sections will describe how to use the Properties Editor.

C-2 PROPERTIES EDITOR

The properties editor for the terminal window control is shown in Figure C-4:

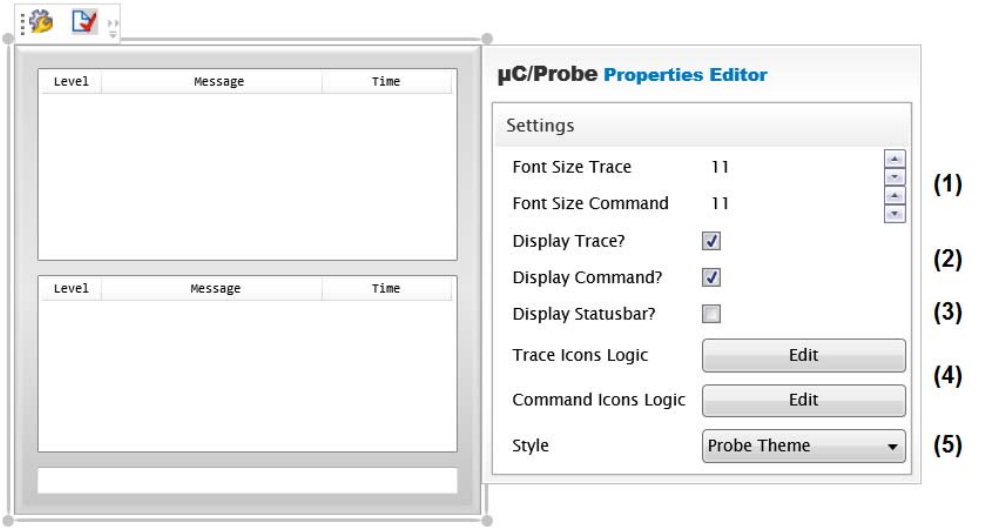


Figure C-4 **Terminal Window Control Properties Editor**

FC-4(1) You can change the font size for each panel.

FC-4(2) You can hide one of the interfaces.

FC-4(3) The terminal window control has a status bar that allows you to see the communication status.

FC-4(4) The icons mapping interface allows you to configure an icon to be displayed whenever a keyword is found in the message back from the target. This is useful if you want to bring more attention to a set of messages, such as warnings or errors.

FC-4(5) The terminal window control is also available in a high contrast theme.

Appendix

D

µC/Trace Triggers Control

µC/Trace is a runtime diagnostics tool for embedded software systems based on µC/OS-III. µC/Trace gives developers an unprecedented insight into the runtime behavior, which allows for reduced troubleshooting time and improved software quality, performance and reliability. Complex software problems which otherwise may require many hours or days to solve, can with µC/Trace be understood quickly, often in a tenth of the time otherwise required. This saves you many hours of troubleshooting time. Moreover, the increased software quality resulting from using µC/Trace can reduce the risk of defective software releases, causing damaged customer relations.

The insight provided by µC/Trace also allows you to find opportunities for optimizing your software. You might have unnecessary resource conflicts in your software, which are "low hanging fruit" for optimization and where a minor change can give a significant improvement in real-time responsiveness and user-perceived performance. By using µC/Trace, software developers can reduce their troubleshooting time and thereby get more time for developing new valuable features. This means a general increase in development efficiency and a better ability to deliver high-quality embedded software within budget.

µC/Trace provides more than 20 interconnected views of the runtime behavior, including task scheduling and timing, interrupts, interaction between tasks, as well as user events generated from your application as shown in Figure D-1. µC/Trace can be used side-by-side with a traditional debugger and complements the debugger view with a higher level perspective, ideal for understanding the complex errors where a debugger's perspective is too narrow.

µC/Trace is more than just a viewer. It contains several advanced analyses developed since 2004, that helps you faster comprehend the trace data. For instance, it connects related events, which allows you to follow messages between tasks and to find the event that triggers a particular task instance. Moreover, it provides various higher level views such as the Communication Flow graph and the CPU load graph, which make it easier to find anomalies in a trace.

µC/Trace does not depend on additional trace hardware, which means that it can be used in deployed systems to capture rare errors which otherwise are hard to reproduce.

Communication Flow
Kernel Service Calls Referencing Any Object (By Service)
CPU Load Graph
Task Accelerometer Task - Execution Time
Task Accelerometer Task - Response Time
Task Startup Task - Execution Time
Task Startup Task - Response Time

Figure D-1 **µC/Trace Analyzer Windows**

The µC/Trace solution consists of three parts:

- The PC application (µC/Trace), used to analyze the recordings as shown in Figure D-1.
- A trace recorder library that integrates with µC/OS-III, provided in C source code.
- Optionally, µC/Probe can be used for the target system connection.

The PC application µC/Trace has been developed for Microsoft Windows.

The trace recorder library stores the event data in a RAM buffer, which is uploaded on request to the host PC using your existing debugger connection or µC/Probe.

And finally, you can use µC/Probe and a special control designed for µC/Trace called µC/Trace Trigger Control, to trigger a recording and launch the µC/Trace analyzer. The µC/Trace Trigger Control is shown in Figure D-2:

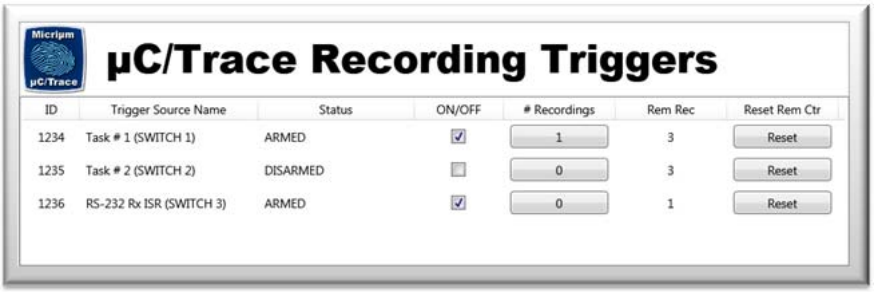


Figure D-2 **µC/Trace Trigger Control**

For more information on µC/Trace, go online to http://micrium.com/tools/uctrace

Appendix

E

Spreadsheet Control

Microsoft® Excel® is a very widely used spreadsheet application developed by the Microsoft® Corporation. It features calculation and graphing tools among other features that now you can embed in your µC/Probe data screens thanks to a technology offered by Microsoft® called *Automation*.

Automation to Excel® allows you to programmatically perform actions such as creating a new workbook, adding data to the workbook, or creating charts. Virtually, all of the actions that you can perform manually through the user interface can also be performed programmatically by using automation.

µC/Probe makes use of automation to allow you to create a workbook, map the spreadsheet cells to your embedded target symbols and make use of all the features Excel® has to offer.

This appendix will show you how to drag-and-drop an instance of Excel® into a µC/Probe data screen and how to associate your embedded target symbols to the spreadsheet's cells.

Microsoft®, Excel® and Windows® are either registered trademarks or trademarks of Microsoft® Corporation in the United States and/or other countries. The use of Microsoft® Excel® automation features by µC/Probe does not imply that Micriμm and/or µC/Probe have any Microsoft® affiliation, sponsorship, endorsement, certification, or approval.

E-1 ADDING AN INSTANCE OF THE SPREADSHEET CONTROL

The Spreadsheet Control is only available on the Professional Edition of µC/Probe. Additionally, you need to have Microsoft® Excel® version 2003 or newer installed on your Windows® PC.

In order to add an instance of Excel®, go to the µC/Probe toolbox in the *Advanced* controls category and drag-and-drop the icon labeled *Spreadsheet* into a data screen as illustrated in Figure E-1:

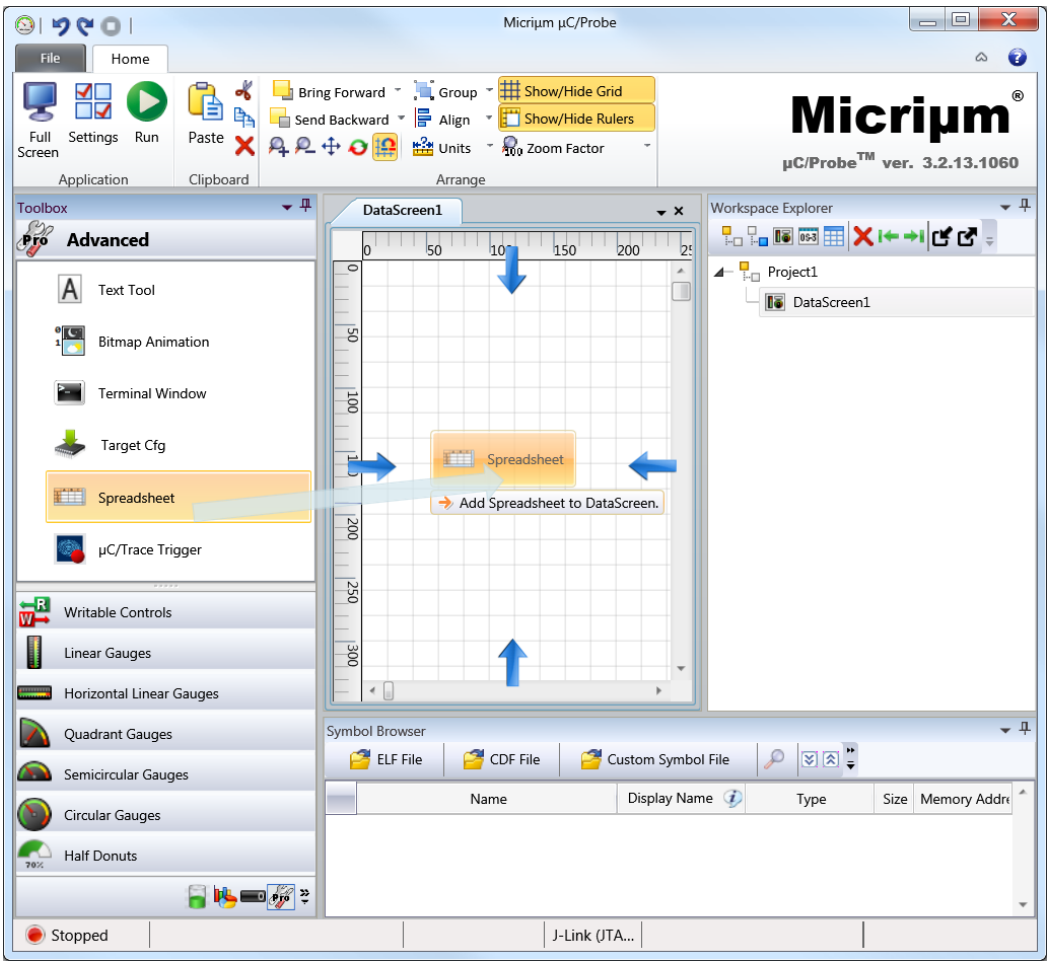


Figure E-1 **Adding an Instance of the Spreadsheet Control**

E-2 CONFIGURING THE SPREADSHEET

Once you have successfully added an instance of Excel® into a data screen, it is time to associate your embedded target symbols from the *Symbol Browser* to any cell in the spreadsheet as illustrated in Figure E-2:

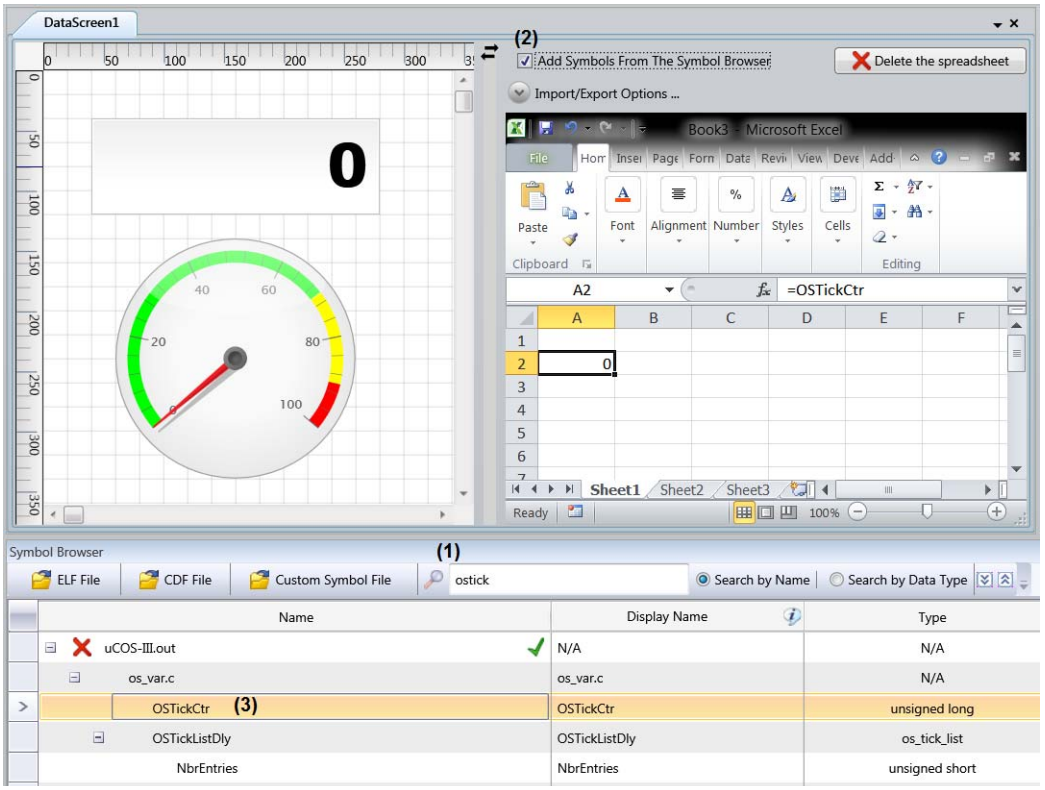


Figure E-2 **Configuring the Spreadsheet**

FE-2(1) You open a symbols file (i.e. ELF file) and browse through it as described in Chapter 3, “µC/Probe Symbol Browser” on page 15.

FE-2(2) Make sure the checkbox labeled *Add Symbols From The Symbol Browser* at the top is ticked and click over the cell that you want to configure.

FE-2(3) Go back to the *Symbol Browser* and double-click over the symbol you want to configure.

E-3 OTHER FEATURES

Figure E-3 shows you other features:

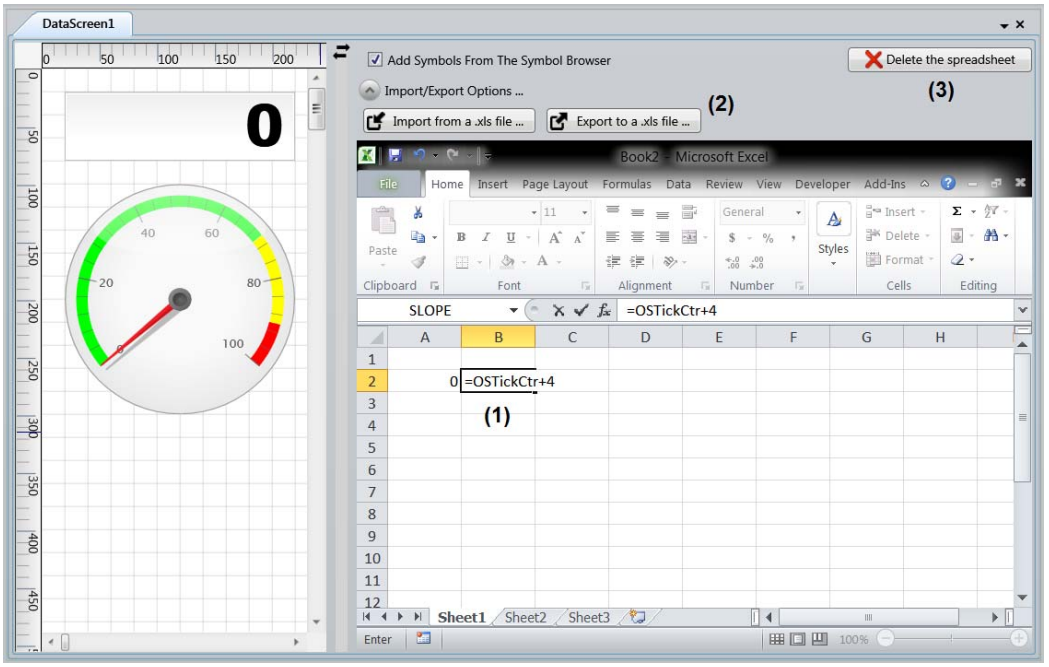


Figure E-3 **Other Features**

FE-3(1) Excel® formulas start by typing in the equal sign in the cell where you want the formula result to appear. As soon as you type in the equal sign, start entering the first few letters of the name of the symbol you want to use in the expression, and for your convenience, Excel® will show you a list of the symbol names you have previously added to the spreadsheet.

When you click on a cell containing a formula in Microsoft® Excel®, the formula always appears in the formula bar located above the column letters.

FE-3(2) You can import and export the spreadsheet along with the embedded target symbols mapping in the form of a regular Microsoft® Excel® file (xls).

FE-3(3) Press the button labeled *Delete Spreadsheet* to remove the spreadsheet from the data screen.

E-4 APPLICATION EXAMPLE

A great way to use µC/Probe and the Spreadsheet Control is to test and calibrate an onboard accelerometer. Imagine your embedded application is reading an accelerometer's X and Y axis and storing them in the global variables `AppAccelX` and `AppAccelY` respectively as shown in Figure E-4:

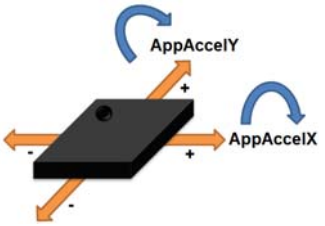


Figure E-4 **Accelerometer**

Very quickly you can add an instance of the spreadsheet control, map two cells to the variables `AppAccelX` and `AppAccelY`, insert a bubble chart from Excel® and there you have a bubble level as shown in Figure E-5:

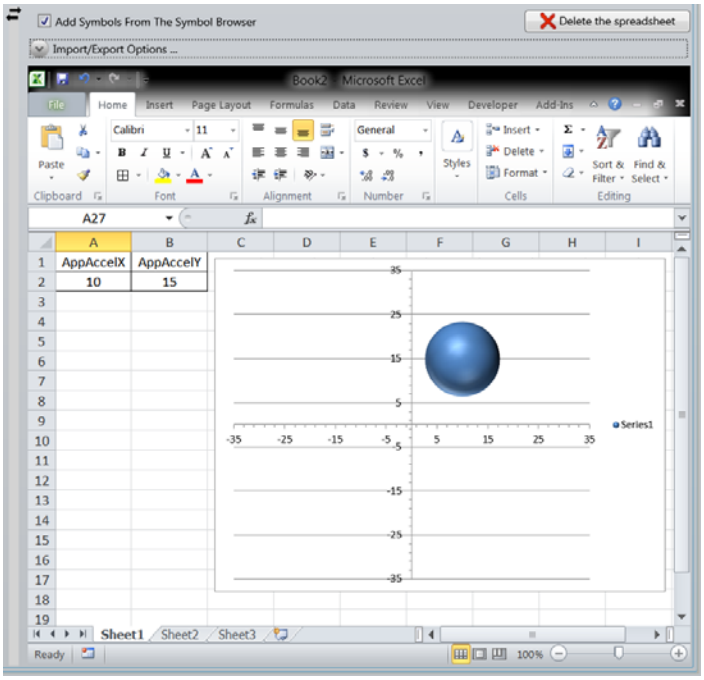


Figure E-5 **Bubble Level Chart**

Appendix

F

Scripting Control

The Professional Edition of µC/Probe comes with a control that helps you programmatically read and write your embedded target symbols on the fly.

You write a script in a C style language to read and write your embedded target symbols and this control will execute your script whenever you want.

Use the µC/Probe Scripting Control to create configuration files, test scripts and to automate the execution of any other common tasks which could alternatively be executed manually by a human operator.

F-1 WRITING A SCRIPT

The µC/Probe Scripting Control supports a programming language that is very similar to C. You write the script as a text file to manipulate variables in two different scopes; the script-side and the embedded target side as described in the following sections.

F-1-1 SCRIPT-SIDE VARIABLES

You declare a variable on the script-side (host PC) by prefixing it with the keyword `var` or with the data type as shown in Code F-1-2:

```
var max_ch = 3;                      /* Declare a script-side variable by using the keyword  */
                                     /* 'var' and initialize it before using it.             */

string lcd_part_nbr = "";            /* Alternatively, you can declare script-side variables */
                                     /* by specifying its data type (char, string, short,    */
                                     /* int, uint, long, ulong, float, double, bool, etc.).  */

int    gain = 10;

string filter_type;                  /* Furthermore, when the data type is specified, you do */
                                     /* not require to initialize the variable.              */

if (max_ch == 8) {
    ...
} else {
    ...
}
```

F-1-2 SCRIPT-SIDE VARIABLES

F-1-3 EMBEDDED TARGET SIDE VARIABLES

The variables on the embedded target side are referenced by prefixing the name of the variable with the dollar sign `$` as shown in the following Code Listing F-1:

```
$AppTimeout   = 100;                 /* Target-side variables are prefixed with the $ sign   */
$AppFrequency =  50;                 /* and they must be present in the target's ELF file.   */

for (int i = 0; i < max_ch; i++) {
    $AppChannels.Ch[i].Gain = gain;
}
```

Listing F-1 **Embedded Target Side Variables**

F-1-4 FLOW CONTROL STATEMENTS

The µC/Probe Scripting Control provides two styles of flow control; branching and looping.

BRANCHING

- `if` statement
- `switch/case` statement
- `?` operator

LOOPING

- `while` loop
- `for` loop
- `do/while` loop

It is your responsibility to ensure the loops have a terminating condition and that the terminating condition can be met. Code Listing F-2 shows an example of using some of these flow control statements:

```
if (max_ch == 8) {
    ...
} else {
    ...
}

switch ($AppSwitches) {
    case 1:
         ...
         break;
}

while ($AppStateCalibrating == true) {
    Sleep("Calibrating", 1000);
}
```

Listing F-2 **Flow Control Example**

F-1-5 BUILT-IN INSTRUCTIONS

Besides the standard flow control statements previously mentioned, the µC/Probe Scripting Control provides a set of built-in instructions to help you create a better user interface.

Code Listing F-3 shows you how to use the `Sleep()` and `Pause()` built-in instructions.

```
for (int i = 0; i < max_ch; i++) {
    $AppChannels.Ch[i].Gain = 0;
                                        /* The Sleep instruction allows you to delay the script */
                                        /* execution (milliseconds) with a custom message.      */
    Sleep("Reaching Steady-State...", 200);
}

$AppSelfTestStart = true;

Pause("Self-Test in progress...");    /* The Pause instruction allows you to pause the script */
                                        /* execution until you press a button in µC/Probe.      */
```

Listing F-3 **Built-in Instructions**

F-1-6 INCLUDING OTHER SCRIPT FILES

Similar to the directive `#include` in the C language, the µC/Probe Scripting Control allows you to include other script files to help you organize your scripts in modules.

When a script file is included, the code it contains inherits the variable scope of the line on which the include occurs. Any variables available at that line in the calling file will be available within the called file, from that point forward. However, all functions defined in the included file have the global scope.

F-2 ADDING AN INSTANCE OF THE SCRIPTING CONTROL

To add an instance of the µC/Probe Scripting Control go to the µC/Probe toolbox in the *Advanced* Controls category and drag-and-drop the icon labeled *Scripting* into a data screen as shown in Figure F-1:

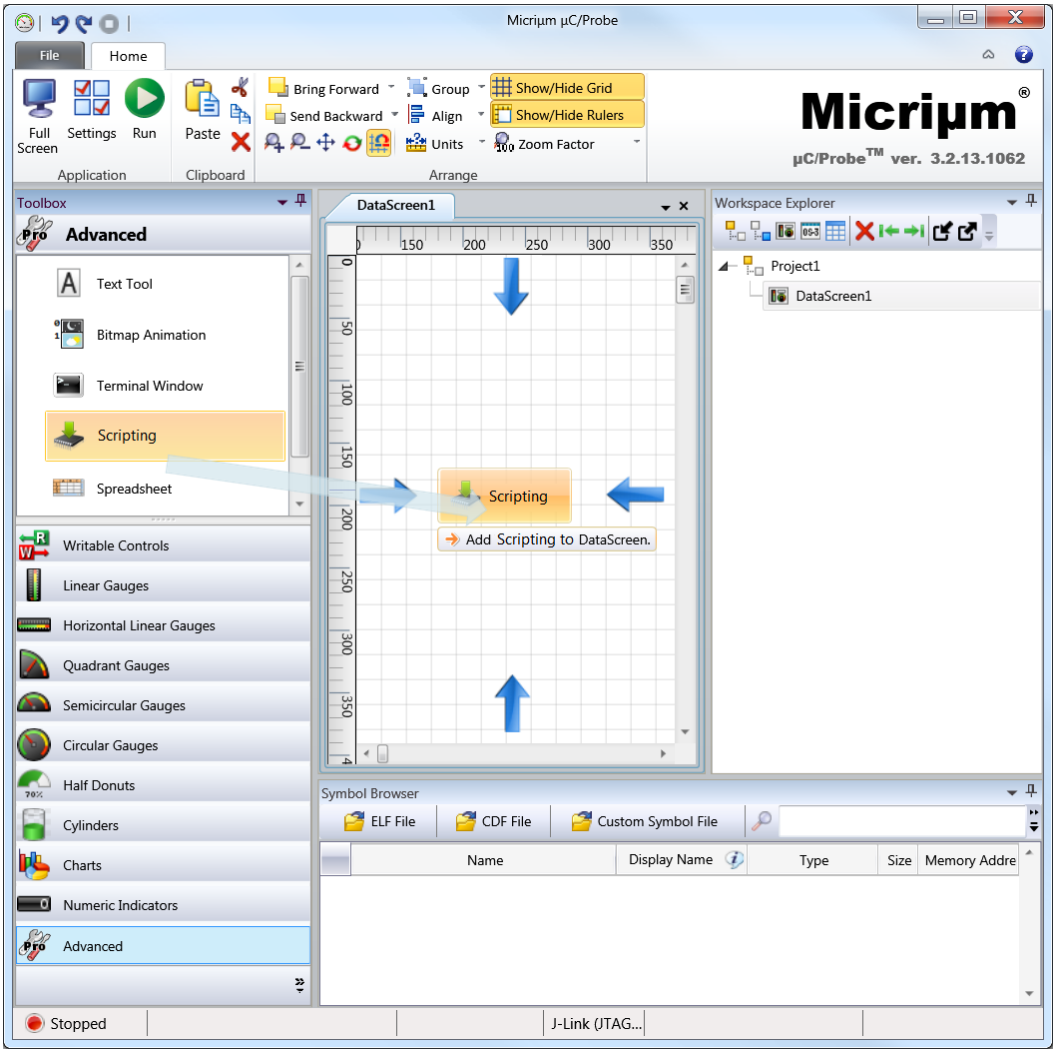


Figure F-1 **Adding an Instance of the Scripting Control**

F-3 CONFIGURING THE SCRIPTING CONTROL

Once you have successfully added an instance of the Scripting Control into a data screen, you have to specify the path to the script file by making click on the properties editor button, which in turn opens a file dialog for you to specify the location of your script file as shown in Figure F-2:

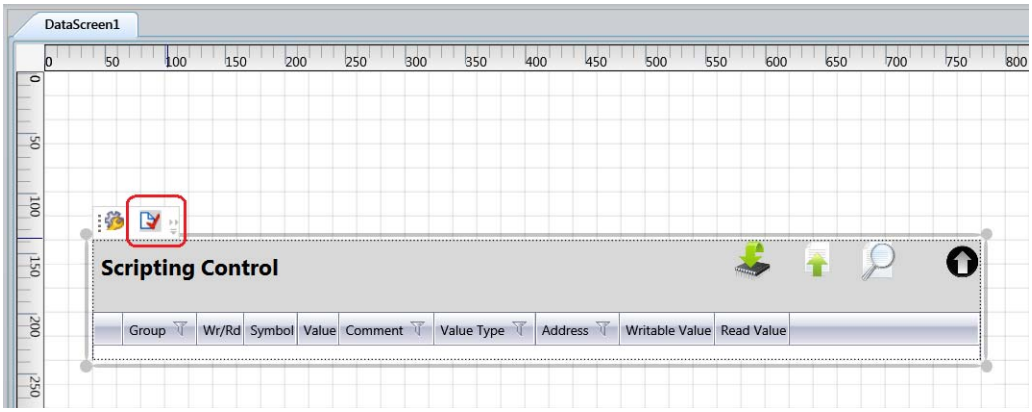


Figure F-2 **Configuring the Scripting Control**

F-4 EXECUTING THE SCRIPT

Suppose you create a configuration script that not only sets the gain of each channel in your data acquisition system, but also activates a system self test and then evaluates the results.

Code Listing F-4 shows the script for such an example.

```
                                    /* Configure the system.                                */
var max_ch = 3;
int gain   = 10;

for (int i = 0; i < max_ch; i++) {
    $AppChannels.Ch[i].Gain = gain;
}

$AppSelfTestStart = true;           /* Gets the system self test started.                   */

                                    /* The Sleep instruction allows you to delay the script */
                                    /* execution (milliseconds) with a custom message.      */
Sleep("Self-Test in progress...", 5000);

                                    /* Evaluate the self test results.                      */
if ($AppSelfTestErr == 0) {
                                    /* The Pause() instruction allows you to suspend        */
                                    /* execution until you press a button in µC/Probe.      */
    Pause("System Configured OK.");
} else {
    Pause("Unable to Configure the System.");
}
```

Listing F-4 **Configuration Script Example**

Once you configure the Scripting Control with the script in Listing F-4, the control will parse the code and will display each of the instructions the control will execute for you as shown in Figure F-3:

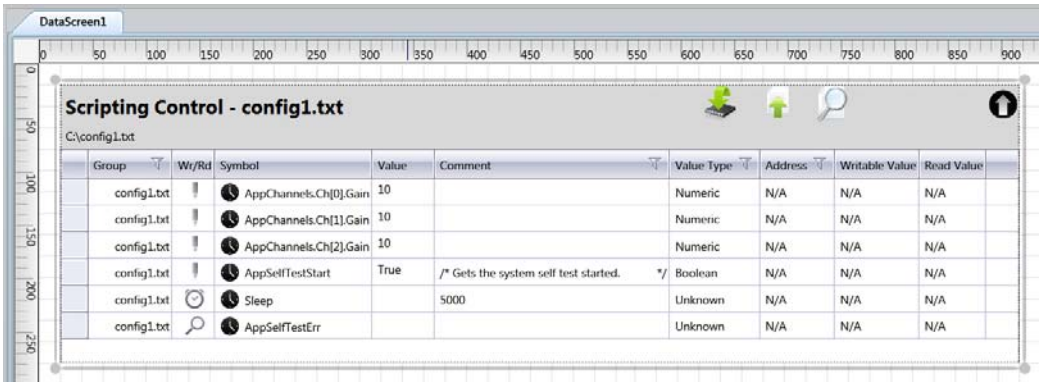


Figure F-3 **Configuration Script Example - Design Time**

When you run µC/Probe, the control will execute your script when you click the button with the chip icon and will display the status of each instruction's execution as shown in Figure F-4:

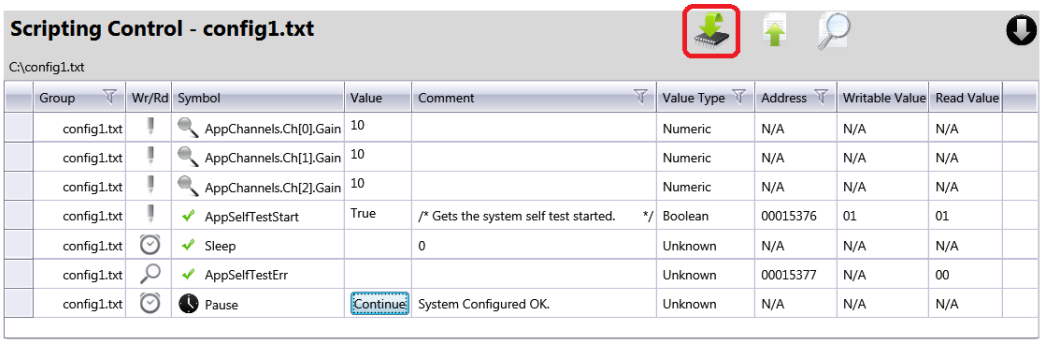

Scripting Control - config1.txt
C:\config1.txt
Group
Wr/Rd
Symbol
Value
Comment
Value Type
Address
Writable Value
Read Value
config1.txt AppChannels.Ch[0].Gain 10 Numeric N/A N/A N/A
config1.txt AppChannels.Ch[1].Gain 10 Numeric N/A N/A N/A
config1.txt AppChannels.Ch[2].Gain 10 Numeric N/A N/A N/A
config1.txt AppSelfTestStart True /* Gets the system self test started. */ Boolean 00015376 01 01
config1.txt Sleep 0 Unknown N/A N/A N/A
config1.txt AppSelfTestErr Unknown 00015377 N/A 00
config1.txt Pause Continue System Configured OK. Unknown N/A N/A N/A

Figure F-4 **Configuration Script Example - Run Time**

Appendix G

Licensing

The Educational Edition of µC/Probe is freely available for students, teachers, and academic organizations. You can purchase µC/Probe for your personally owned computer or those owned by your employer by either a perpetual license or a subscription based license as follows:

- Monthly Subscription License

  Our e-commerce partner FastSpring will bill you automatically each month and you may cancel the subscription at any point, no questions asked.

- Yearly Subscription License

  Our e-commerce partner FastSpring will bill you automatically each year and you may cancel the subscription at any point, no questions asked.

- Perpetual License

  Our e-commerce partner FastSpring will bill you only once and you will own the license for an unlimited period of time.

Note: Keep in mind that all µC/Probe sales are final and non-refundable. The Educational Edition of µC/Probe is available for free to enable you to "try before you buy".

For example, if you purchase a yearly subscription and wish to cancel the subscription after a couple of months, we will cancel your subscription and any future billing. However, we will not be able to make any partial refunds and you would still be charged for the first year's subscription.

G-1 ORDERING

If you decide to purchase a license of µC/Probe you can go to our online store at:

http://micrium.com/tools/ucprobe/buy

The table below summarizes the main differences among the different editions of µC/Probe:

Feature	**Educational Edition**	**Basic Edition**	**Professional Edition**
Design Mode	x	x	x
Run-Time Mode (timeout in minutes)	x (1)	x (no-timeout)	x (no-timeout)
Maximum Number of Data Screens	1	unlimited	unlimited
Maximum Number of Gauge Styles	5	unlimited	unlimited
Maximum Number of Numeric Indicator Styles	3	unlimited	unlimited
Thermometer	x	x	x
Cylinder Indicators	x	x	x
Button Controls	x	x	x
Slider Controls	x	x	x
Bit Control	x	x	x
Marker Chart	x	x	x
Line Chart	x	x	x
Area Chart	x	x	x
Scatter X-Y Chart			x
µC/OS-III Kernel Awareness	x	x	x
Terminal Window Control			x
Scripting Control			x
Microsoft® Excel® Bridge			x
µC/Trace Trigger Control			x
Import/Export Data Screens		x	x

Table G-1 **µC/Probe Editions Comparison Table**

G-2 ACTIVATING

Once you place an order on our online store you will receive an e-mail message with your license key.

To activate your copy of µC/Probe, you need to have Internet access. Click on *File* -> *Activation* and a window similar to the one shown below will be displayed:

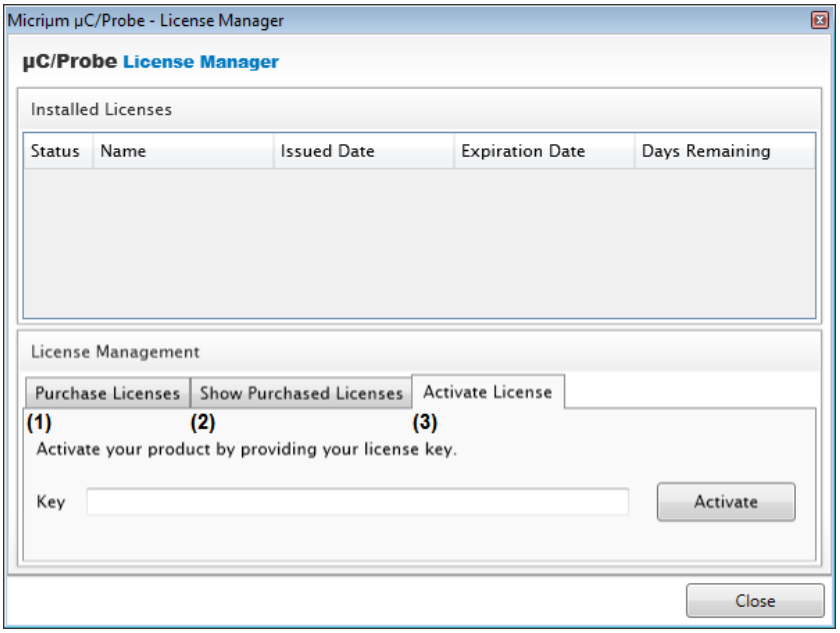


Figure G-1 **License Manager**

FG-1(1) Use this tab to go online and purchase a license of µC/Probe.

FG-1(2) Use this tab to obtain a list of all your purchased licenses. The list will be sent via e-mail to the e-mail address registered when you placed the order. All you need to do is provide the invoice number.

FG-1(3) Use this tab to activate a license key and node-lock it to your computer. All you need to do is copy and paste the license key from the e-mail message you received and press the button *Activate*.

Please contact us for further information about pricing, ordering options, license activation or cancelling a subscription at:

Micriμm
1290 Weston Road, Suite 306
Weston, FL 33326

+1 954 217 2036
+1 954 217 2037 (FAX)

E-Mail : sales@micrium.com
Website : www.micrium.com

Appendix H

Bibliography

- Labrosse Jean. *µC/OS-II The Real-Time Kernel*. R&D Technical Books, ISBN 1-57820-103-9, 2002.
- Labrosse Jean. *µC/OS-III The Real-Time Kernel*. Micriµm Press, ISBN 978-0-98223375-3-0, 2009.
- Légaré Christian. *µC/TCP-IP The Embedded Protocol Stack*. Micriµm Press, 2011.

Index

A

animation 50
animation properties editor 50–51
area charts 60

C

chart properties editor 61
chart series editor 63
charts 34, 60
charts properties editor 61
charts series editor 63
checkbox properties editor 55
circular gauges 33
communication settings 21–22
  J-Link 23
  RS-232 26
  TCP/IP 24–25
configuration
  terminal window control 74
custom slider
  example 53
  properties editor 53
custom switch
  properties editor 54
cylinders 34

D

data flow 6
design time 11

E

ELF file 15
  browsing 16
example 38

F

formatting
  properties editor 48

G

general settings 20

H

half donuts 33
horizontal linear gauges 31

I

indicators 39

J

J-Link 23

K

kernel awareness screen
  miscellaneous 70
  task list 71

L

layout design tools 36
line charts 60
linear gauges 31

M

marker charts 60
message URL mailto
  sales@micrium.com?subject=uC/USBD for uC/Probe Information 21
miscellaneous (tools) 35

N

numeric indicator
  properties editor 50
numerics (tools) 35

O

ordering 80, 84, 93
overview 9

P

properties editor

animation 50–51
checkbox 55
custom slider 53
formatting 48
numeric indicator 50
push button 56
repeat button 58
slider control 52
terminal window control 75
toggle button 57
push button
properties editor 56

Q

quadrant gauges 32

R

range and colors editor 49
repeat button
properties editor 58
RS-232 26
run-time mode 43
run-time mode checklist 42

S

Segger J-Link 23
semicircle gauges 32
slider control
properties editor 52
status bar 43
symbol browser
loading an ELF file 15
symbols grouped by C file 16
symbols 39
symbols manager 40

T

TCP/IP 24–25
terminal window control 72–74
configuration 74
properties editor 75
toggle button
properties editor 57
toolbar settings 19
toolbox 29
charts 34
circular gauges 33
cylinders 34
half donuts 33
horizontal linear gauges 31
linear gauges 31
miscellaneous 35
numerics 35
quadrant gauges 32
semicircle gauges 32
writable controls 30
trace triggers control 76–77

V

virtual controls 39, 47, 52
virtual indicators 47–48

W

Windows application 13
workspace explorer 27–28
writable controls 30

Z

µC/Probe data client 11
design time 11
µC/Trace triggers control 76–77